To assure the highest quality computer instruction for your students you could hire a team of computer programmers, system analysts and designers, computer technicians, computer educators, graphic artists, writers, editors and professional production personnel.

Or...

The VideoSoft Series is a remarkable breakthrough in computer education. Over a year in development, this color videotape library teaches BASIC (Beginners All-purpose Symbolic Instruction Code) computer programming by using a step-by-step method that is at once interesting and easy to understand.

At the completion of the VideoSoft course the student will have achieved a solid base of fundamental computer skills.

VideoSoft™ Instructional Programming Series

What is the VideoSoft advantage?

The VideoSoft Series is designed to be used by instructors already on staff. Teachers with or without programming skills can preview each lesson and present the videotaped instruction. Despite fiscal restraints or the lack of trained personnel, school districts can meet the increasing need for computer skills at the junior and senior high school level.

Schools with limited computer time for students will find the VideoSoft course especially valuable since all concepts are demonstrated on a computer during the videotaped lesson.

Why videotape?

Videotape offers the unique advantage of showing exactly what a student will see when seated at a computer terminal. In addition, the videotape instructor is able to use visual illustrations and three-dimensional representations of computer logic. The videotaped lessons assure the school district of sound and consistent instruction in the fundamentals of programming in all classes. The videotaped lessons can be paused or replayed for review of any concept.

What is the VideoSoft instructional model?

The VideoSoft instructor carefully builds a solid base of knowledge and practical application. Every function and command is explained and illustrated on the TV screen; graphic aids are used where necessary to help students understand the concepts presented. The information is put into practice so that students can see exactly what the command will do if written correctly or how it will "crash" if written improperly. No new information is given until each preceding concept has been firmly established. Although the pace is fast, nothing is omitted on the assumption that the student already knows the information.

What is contained in the student study guide and teacher's guide?

The Student Study Guide is a 134 page book designed to enable the student to review and apply new programming concepts used by the VideoSoft instructor.

Each lesson in the Study Guide includes lesson objectives and a short description of the lesson, followed by a complete listing of the programs used in the lesson, an explanation of the lesson assignment, two additional assignments and a self-check test.

The appendices at the back of the Study Guide include answers to the self-check tests, a listing of reserved words, a glossary of terms and commands, and other reference materials.

Each lesson's assignment focuses on the concepts taught in the lesson. Each lesson includes two additional assignments: one for review of the lesson concept and one which encourages the student to use the concept creatively.

The 260 page Teacher's Guide includes the entire text of the student guide along with additional information necessary to evaluate the students' understanding and expand on the information provided in each lesson.

The Teacher's Guide includes detailed explanations of the answers to each assignment and additional assignment and a complete program listing of the VideoSoft answer to each assignment and additional assignment.

The teacher's materials include seven evaluation tests to periodically evaluate the students' understanding of the concepts and commands. Answer keys to the evaluation tests are also provided.

The Teacher's Guide includes general recommendations for the use of the course in different educational environments and specific notes on each lesson to further explain or expand on the concepts presented.

Lesson Outlines

To insure that the VideoSoft course is as comprehensive as possible, the Series covers not only the general concepts of BASIC but also the specific implementation of the different versions of BASIC used on specific computers.

Each VideoSoft Instructional Series includes lessons covering the following concepts:

Introduction:

A brief history of computers.
Introduction to the computer. Introduction to BASIC programming, line numbers, the PRINT
command, LISTing and RUNning programs.

Programming in BASIC:

SAVE AND LOAD commands. Other diskette commands. Documenting programs with REMarks. Inserting and deleting lines. (INVERSE, NORMAL, FLASH and SPEED commands for Apple. AUTO, EDIT and RENUMber commands for MBASIC.)

Variables:

The concept of a variable. Naming variables. String and numeric variables. Variables with LET, PRINT and INPUT.

Math calculations:

BASIC math operators. Multiple calculations and math hierarchy (priorities of math calculations). E-Notation.

Formatted printing:

Use of commas and semicolons. The TAB command. (HTAB and VTAB for Apple. PRINT@ for TRS-80. LOCATE for IBM PC, PC *jr.*)

Conditioning commands:

GOTO command. IF-THEN command pair. Comparisons, equal, greater than, less than, not equal to. Multiple comparisons, AND, OR. (ELSE for MBASIC.)

Programming techniques, part I:

Shortcuts and multiple statement lines. Programming in modules using program outlines and logic paragraphs.

Loops:

Concept of a loop. Loops with GOTO. Loops with IF-THEN. The FOR-NEXT command pair. The STEP command. The concept of loop "nesting."

Subroutines:

Concept of a subroutine. The GOSUB-RETURN command pair. The ON-GOSUB and ON-GOTO command pairs. BASIC FLOWCHARTING.

Arrays:

The concept of an array. Elements and subscripting. DIMensioning arrays. Arrays and loops. The DATA-READ command pair. Multi-dimensional arrays. Multi-dimensional arrays and nested loops.

String functions:

Manipulating character strings with LEN, LEFT$, RIGHT$, MID$. Concatenation. Converting variables with VAL AND STR$. Loops and strings.

Data files:

Concept of a data file. SEQUENTIAL DATA FILES. Opening and closing data files. Storing, retrieving, adding, and updating data. RANDOM ACCESS DATA FILES. Concept of a record. Opening and closing random access files. Storing, retrieving, adding, and updating data.

Programming techniques, part II:

Special programming and debugging techniques. Interpreting programs written by others. Computers as logical tools. Course conclusion.

Lo resolution graphics (Apple only):

Concept of Lo Res color graphics. Color names and numbers. Column and row coordinates. The graphics commands: GR, COLOR, PLOT, TEXT, HLIN, VLIN, and SCRN. Drawing pictures and changing colors.

Concepts of animation (Apple only):

Creating the illusion of movement by drawing and "erasing." Animation over a solid background. The RND command and the INT command. PEEK and POKE commands. Animation over any multicolored background.

Advanced screen formatting (MBASIC: TRS-80, IBM PC, and IBM PCjr):

The PRINT USING command, ASCII, and special character sets.

Advanced data files (MBASIC: TRS-80, IBM PC, and IBM PCjr.):

Concept of a field. Numeric data types, integer, single precision, and double precision. Structuring a record with fields.

VideoSoft™ Instructional Programming Series

What materials are included in the VideoSoft course?

The VideoSoft Instructional Series in BASIC Computer Programming includes:

- **14 Lesson Tapes and 2 Answer Tapes.**
  Each of the 14 Lesson tapes is approximately 40 to 50 minutes in length. The answers to the assignments are included on two separate tapes with answers varying from 2 to 15 minutes, depending on the complexity of the assignment.

  Each of the tapes is individually packaged in a plastic library case. The series comes in a plexiglass "bookshelf" for easy and safe storage.
- **A simulated computer keyboard.**
  This simple aid assists the student particularly in the early lessons to become familiar with the computer keyboard if computers are not immediately available.
- **A 134 Page Student Study Guide.**
  The Study Guide is comb-bound in a soft cover and serves as a resource for preview and review of the lessons, concepts and commands. Study Guides also contain full listings of all programs used in the course.
- **A 260 Page Teacher's Guide.**
  The Teacher's Guide is bound in a three ring hardcover notebook and includes a complete copy of the Study Guide and a wealth of additional materials, such as detailed explanations of all assignment answers, additional information to extend the lessons if desired, evaluation tests and answers.

What type of equipment will be needed?

A videotape player and television monitor are needed to show the tapes. VideoSoft can provide whatever size or type tape you need.

What if tapes get worn or ruined?

VideoSoft will replace all worn or damaged tapes for a period of three years for cost plus postage and handling.

What is the cost of the series?

Videotape computer courses for business and industry run in excess of $700 per hour. The VideoSoft Series has been held below $200 per hour to allow schools and school districts greater access to this revolutionary breakthrough in computer programming instruction.

The course, including the taped lessons, answers, storage case, one Teacher's Guide and one Study Guide, is available in three videotape formats:

VHS or BETA format $2,378.00
¾ inch Umatic format $2,538.00
Additional Teacher's Guides . . . $12.50 each
Additional Student Study Guides $10.00 each

For information concerning volume discounts and closed circuit broadcast rights, please contact the VideoSoft offices.

How can I get more information?

VideoSoft can provide a demonstration videotape and sample Teacher's Guide free of charge. All we ask is that you return the videotape when you have finished with it. We will pay the postage both ways. Just fill out the attached business reply card or contact us at the VideoSoft offices.

- **A 134 Page Student Study Guide.**
  The Study Guide is comb-bound in a soft cover and serves as a resource for preview and review of the lessons, concepts and commands. Study Guides also contain full listings of all programs used in the course.
- **A 260 Page Teacher's Guide.**
  The Teacher's Guide is bound in a three ring hardcover notebook and includes a complete copy of the Study Guide and a wealth of additional materials, such as detailed explanations of all assignment answers, additional information to extend the lessons if desired, evaluation tests and answers.

What type of equipment will be needed?

A videotape player and television monitor are needed to show the tapes. VideoSoft can provide whatever size or type tape you need.

What if tapes get worn or ruined?

VideoSoft will replace all worn or damaged tapes for a period of three years for cost plus postage and handling.

What is the cost of the series?

Videotape computer courses for business and industry run in excess of $700 per hour. The VideoSoft Series has been held below $200 per hour to allow schools and school districts greater access to this revolutionary breakthrough in computer programming instruction.

The course, including the taped lessons, answers, storage case, one Teacher's Guide and one Study Guide, is available in three videotape formats:

VHS or BETA format $2,378.00
¾ inch Umatic format $2,538.00
Additional Teacher's Guides . . . $12.50 each
Additional Student Study Guides $10.00 each

For information concerning volume discounts and closed circuit broadcast rights, please contact the VideoSoft offices.

How can I get more information?

VideoSoft can provide a demonstration videotape and sample Teacher's Guide free of charge. All we ask is that you return the videotape when you have finished with it. We will pay the postage both ways. Just fill out the attached business reply card or contact us at the VideoSoft offices.

For a free demonstration videotape and sample Teacher's Guide please fill out the attached business reply card or contact us at the VideoSoft offices.

I am interested in the VideoSoft Instructional Programming Series. Please send me a demonstration tape, free of charge. I will return it when finished and understand that Videosoft will pay all postage costs.
Please fill in the following:
Are you currently using computers in the classroom?_____
If yes, what brand?_____
If no, what brand are you considering?_____
How are the computers being used or how will they be used?
☐ computer assisted instruction ☐ literacy ☐ programming
☐ other_____
Number of junior high and high schools in your district?_____
What kind of videotape player will you be using for the demonstration tape?
☐ ½" Beta ☐ ½" VHS ☐ ¾" U-Matic

Name_____
Position_____
School District_____
Address_____
City_____ State_____ Zip Code_____
Tel. ( )_____
Please allow two weeks for delivery.

deoSoft™ ructional gramming Series

uls are included in the urse?

uctional Series in BASIC Computer Program-

and 2 Answer Tapes.

sson tapes is approximately 40 to 50 mi-
The answers to the assignments are included
tapes with answers varying from 2 to 15 mi-
on the complexity of the assignment.

is individually packaged in a plastic library
comes in a plexiglass "bookshelf" for easy and

nputer keyboard.

ssists the student particularly in the early les-
amiliar with the computer keyboard if com-
mediately available.

For a free demonstration videotape and sample Teacher's Guide please fill out the attached business reply card or contact us at the VideoSoft offices.

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

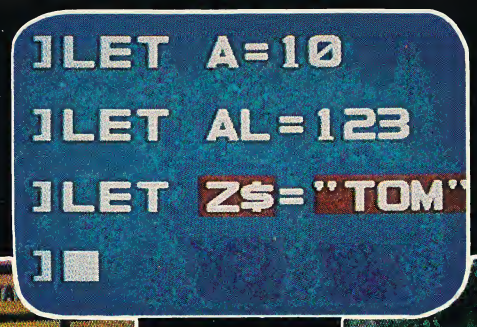
BUSINESS REPLY CARD
FIRST CLASS PERMIT NO 45 LITTLE ROCK, AR

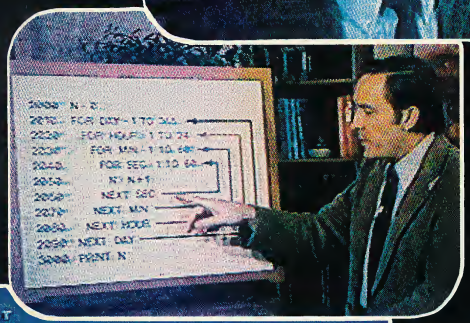
POSTAGE WILL BE PAID BY ADDRESSEE

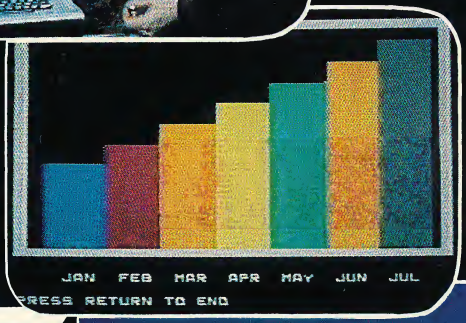
VIDEOSOFT™
2101 SOUTH BROADWAY
LITTLE ROCK, ARKANSAS 72206-9990

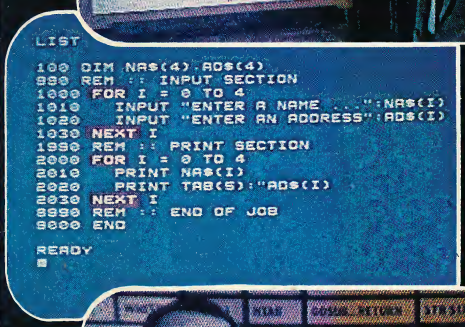
You could use VideoSoft™

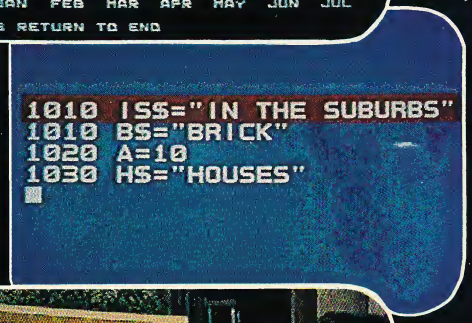
The VideoSoft™ Advantage

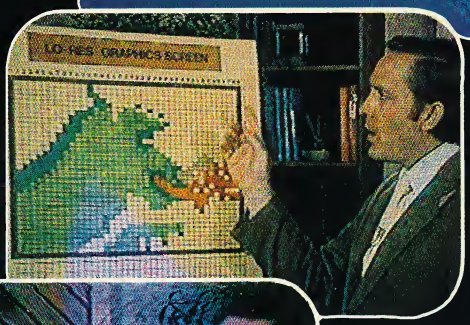


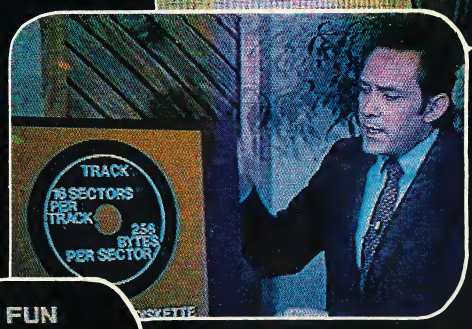





LIST
100 DIM NA$(4),AD$(4)
990 REM :: INPUT SECTION
1000 FOR I = 0 TO 4
1010 INPUT "ENTER A NAME ...";NA$(I)
1020 INPUT "ENTER AN ADDRESS";AD$(I)
1030 NEXT I
1990 REM :: PRINT SECTION
2000 FOR I = 0 TO 4
2010 PRINT NA$(I)
2020 PRINT TAB(5);AD$(I)
2030 NEXT I
9990 REM :: END OF JOB
9000 END
READY







VideoSoft
2101 South Broadway
Little Rock, Arkansas 72206
(501) 376-2083

May 23, 1985

Sister Mary Lucille
Principal
Holy Name School
1007 S W Blvd.
Kansas City, KS 66103

Dear Sister Lucille:

I enjoyed meeting you at the NCEA Convention in St. Louis. I hope your time there was productive. I have enclosed the VideoSoft™ package you requested. The demonstration videotape, Teacher's Guide, price and ordering information provide a comprehensive explanation of the VideoSoft Series. The enclosed brochure provides an abbreviated overview of the Series.

The enclosed Lesson Nine, "Arrays", from the Instructional Series for the Apple computer reflects the teaching techniques and quality instruction consistent throughout the Series.

The clarity of the instruction makes the VideoSoft Series effective even for students with limited or no computer background. The VideoSoft instructor explains each command and programming concept using visual aids where appropriate. A BASIC program is then used to demonstrate each concept. VideoSoft's SimComp™ animation technique creates a computer CRT that is very interesting and readable. The computer programs are enlarged, highlighted or animated as needed to clarify each command and concept.

For beginning students, the VideoSoft course is most frequently used in a lecture-lab environment which allows the school to maximize the use of teachers and computer lab time. For more advanced students, the Series is generally used for independent study or as a teaching aid to explain the commands and concepts of BASIC programmming.

The written materials support and enhance the videotape lessons. We have enclosed the Teacher's Guide, which includes the entire text of the Student Guide as well as answers to the assignments, evaluation tests, teaching suggestions, and lesson extensions.

2101 South Broadway, Little Rock, Arkansas 72206 • (501) 376-2083

The VideoSoft guarantee allows replacement of all worn or damaged tapes for a period of three years for cost plus postage and handling. In addition, if you are not completely satisfied with the Series, it may be returned anytime within the first 30 days for a full refund. Please note that we provide, at no additional charge, 30 Student Guides as well as a Teacher's Guide and program listings on diskette with every Series purchased.

We offer a review time of ninety (90) days in which to examine the Series. When you have completed your review of videotapes and Teacher's Guide, return them using the prepaid return address label enclosed. If you require more than 90 days to review the Series, simply notify us and we will be more than happy to extend your review time.

For information on duplication and broadcast rights, quantity discounts and terms, refer to the VideoSoft Order form.

I believe you will share our excitement about this remarkable approach to computer programming instruction. If you have any questions after reviewing the enclosed material, please contact me at the VideoSoft offices.

Sincerely,

Randy Block

Randy Block
Executive Vice President

RB/ec
ENCL.

December 18, 1985

Sister Mary Lucille
Principal
Holy Name School
1007 Southwest Boulevard
Kansas City, KS 66103

Dear Sister Lucille:

In May of this year, you requested a demonstration tape of our VideoSoft Series. Because of the considerable expense in providing these tapes to our prospective customers, we asked that you return the tape to us when you had finished viewing it, and we enclosed postage for your convenience in doing so.

Have you yet had an opportunity to view the tape? If so, may we ask that you return it to us at your earliest convenience so that we may respond to similar requests from other school systems.

If you need more viewing time, or if we can provide you with additional information to assist you in making a purchase decision, please contact me at the address below.

Thank you for your consideration.

Yours Truly,

Marla Johnson

Marla Johnson
Controller

MJ/ch

2101 South Broadway, Little Rock, Arkansas 72206 • (501) 376-2083

HOW TO PREVIEW VIDEOSOFT

The demonstration videotape and Instructor's Guide which you are about to review are typical of the entire VideoSoft BASIC Programming Series. We have chosen Lesson Nine, "Arrays", because it demonstrates the instructional techniques used throughout the course and illustrates how VideoSoft can make even the "complicated" concepts of arrays easy to understand and use. To assist you in your evaluation of this outstanding educational product, we recommend that you review the product as follows:

GENERAL PREPARATION

- **READ: Teaching Suggestions for Lesson Nine.**
  Student Guide Overview for Lesson Nine.

- **SCAN: Lesson Nine Computer Programs in Student Guide.**
  Lesson Nine Self-Check Tests in Student Guide.
  Student Guide Overviews for Lessons One through Eight.
  Teaching Suggestions for Lessons One through Eight.

By the time your students view Lesson Nine, they will have written between eight and twenty-four programs, depending on how many additional assignments were given. By Lesson Nine, your students would already be familiar with the concepts of variables, IF-THEN statements, FOR-NEXT loops, subroutines and general program structure. However, the student will find Lesson Nine easy to follow regardless of his or her background.

GROUP VIEWING

If several members of your staff are to be involved in the evaluation process, we recommend that the lesson be viewed as a group, followed by an open discussion. This will allow the staff members involved to share impressions of the course. Ideally, enough time should be allowed to permit each person to preview the written materials prior to viewing.

If a group viewing is not practical, then we recommend that time be set aside for a group evaluation after all participants have had the opportunity to review the Instructor's Guide and the videotape.

AFTER THE REVIEW

After you have completed your review, you may have additional questions such as how the VideoSoft course fits into your curriculum or how it is being used in other schools throughout the United States. We encourage you to contact one of our staff members to discuss the integration of VideoSoft into your school or for any other additional information.

ABOUT THE VIDEOTAPE

Because your demonstration tape has possibly been exposed to extremes of heat or cold during mailing, it will perform best if left at room temperature for approximately 24 hours before viewing. Although all of our videotapes are subjected to rigid quality control standards, the demonstration tape you have received may have been viewed dozens of times and traveled many miles. If your demonstration tape appears worn or damaged, simply return it to us, and we will send you a new tape.

We look forward to serving you.

Sincerely,

Randy Block

Randy Block, Executive Vice-President
VideoSoft, Inc.

RB/ec

2101 South Broadway
Little Rock, Arkansas 72206

Chris Gruber

Sister Mary Lucille
Principal
Holy Name School
1007 Southwest Boulevard
Kansas City, KS 66103

Received [illegible]

16 Tapes - Tapes
(2½ hrs ea
30 Lesson books
w/ instructions
& Guitar Chart

FROM:

SPECIAL
4th CL. RATE.

VIDEO JOE
TM

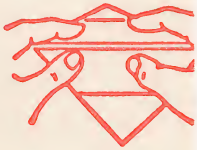
2101 South Broadway
Little Rock, Arkansas 72206

N PEEL Plus™

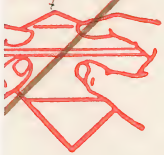
Fasson®

CRACK'N PEEL Plus™

PATENT PENDING



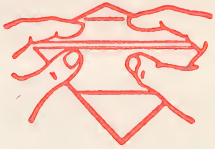
lus™



Fasson®

CRACK'N PEEL Plus™

PATENT PENDING



Fa

CRA

PATENT PEN

Fasson®

CRACK'N PEEL Plus™

PATENT PENDING

Fasson®

Videosoft™ Instructional Programming Series

Applesoft Basic

TEACHER'S GUIDE

VideoSoft™ Instructional Programming Series for Applesoft™ Basic

Copyright 1983 by *VideoSoft*, Inc.
All rights reserved. With the exception of the Evaluation Tests which may be reproduced for testing purposes, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

Manufactured in the United States of America

®APPLE is a registered trademark of Apple Computer Inc.

VideoSoft™ Instructional Programming Series For Applesoft™ BASIC

TABLE OF CONTENTS

I. Introduction T1
II. Course Objectives T2
III. Video Tape Library T2
IV. Student Study Guide Overview T2
V. Instructional Plan T3
VI. Teaching Suggestions
A. General Teaching Suggestions T7
B . Lesson Notes T8
VII. Student Study Guide 1-132
VIII. Lesson Assignment Answers T19
IX. Additional Assignment Answers T54
X. Evaluation Tests T96
XI. Evaluation Test Answers T116

TEACHER'S EDITION

The Teacher's Edition contains the entire text of the Student Study Guide along with additional materials which enable the teacher to expand and further explain the concepts and commands and to evaluate the students' progress.

The Teacher's Edition includes the following:

- The *Course Objectives* express the goals of the series.
- The *Video Tape Library* section explains the physical contents of the VideoSoft™ Instructional Series.
- A brief explanation of the contents of the *Student Study Guide* is provided.
- The *Instructional Plan* includes recommendations for the presentation of the course in various educational environments, recommendations for the evaluation of student progress, and recommendations for enhancing and expanding the students' course.
- The *Teaching Suggestions* provide further explanation and expansion of concepts where appropriate. The suggestions include some general comments concerning the course, followed by notes on each lesson.
- The *Answers to the Assignment* and *Additional Assignments* are found in the back of the Teacher's Edition.
- *Evaluation Tests* and answers are provided following the *Answers* section.
- A *Simulated Keyboard* is enclosed in the folder pocket to help familiarize the students with a standard computer keyboard.

COURSE OBJECTIVES

1. To provide the student with the opportunity to learn the fundamental concepts and commands necessary to program a computer in BASIC.
2. To instill programming habits and techniques which reflect a clear and structured thinking process.
3. To provide the student with a solid foundation in computer programming which can be utilized in advanced programming applications.

THE VIDEO TAPE LIBRARY

The video tape library consists of 14 lesson tapes and two assignment answer tapes. The lesson tapes are approximately forty to fifty minutes long and consist of a series of lectures with step by step instruction covering the fundamental concepts and commands necessary to program a computer in BASIC.

Each lesson tape contains an introduction to the lesson, the lesson itself, a summary of the lesson, and an assignment to be completed before viewing the next lesson tape.

The lecture utilizes visual aids to illustrate important concepts and then demonstrates the concepts and commands on the computer so the student can learn how to enter programs and how the computer interprets the instructions it is given.

The two answer tapes contain the answer to each assignment in sequential order. Answer Tape One contains the answers for lessons one through eight. Answer Tape Two covers the answers for lessons nine through thirteen. Lesson fourteen, the final lesson of the course, has no assignment.

For the student, the video taped instruction provides a clear and concise explanation of the concepts and commands needed to learn to program in BASIC. Furthermore, the programs used throughout the course provide the student with an excellent model for good programming habits.

Video taped instruction offers many advantages to the teacher. Since the *VideoSoft* instructor has the primary lecture responsibility, the teacher is free to concentrate his or her creative talents on enhancing the students' understanding or assisting slower learners.

Teachers with limited computer knowledge are encouraged to take the *VideoSoft* course before attempting to teach it. This provides teachers with a knowledge of the fundamentals of computer programming, placing them in a position to be of greater assistance to the students.

The video tapes provide an additional advantage for teacher and student in that the tapes can be paused and a concept discussed, or the tapes can be rewound and portions of the lecture replayed for review.

The 14 lesson tapes and two answer tapes are packaged in a plexiglass storage case. In addition, each video tape is individually packaged in a plastic album cover. If a video tape is damaged, please contact the *VideoSoft* office concerning replacement policies.

The APPLE CRT signal produces a momentary break-up in the video signal when switching from the APPLE CRT segments to standard video. This is due to a difference in the scan lines of the APPLE CRT and standard video signal. The one second "rolls" in the video are momentary and are not the result of a malfunction of either the video tape or your tape player.

STUDENT STUDY GUIDE OVERVIEW

The Student Study Guide has been designed to enable the student to review and apply new programming concepts and to help the student easily follow the program examples used by the *VideoSoft* instructor. A course outline can be found at the beginning of the Student Guide.

Every lesson in the Study Guide contains the following:

- a short description of the lesson content and objectives.
- a complete listing of the demonstration programs used in the video taped presentation.
- Self-Check Tests which cover the materials introduced in the lesson enable the student to both review and check his/her understanding of the concepts covered.
- a copy of the assignment given by the video tape instructor to be completed prior to the viewing of the next lesson. Lesson assignments are designed to provide students with the opportunity to apply new and previously explained programming concepts and techniques.
- additional assignments which afford the student the opportunity to further practice, extend, and/or review course material. The highly motivated student may complete them on his/her own, or they may be assigned by the teacher.
- appendices which contain the answers to all the Self-Check Tests, a listing of all reserved words, a color number chart for lo-resolution graphics, an explanation of scientific notation, a complete glossary of terms and summary of commands, and a sample lo-resolution screen grid to plot graphics on the computer.

INSTRUCTIONAL PLAN

The Instructional Plan consists of three sections: recommendations for the presentation of the course, recommendations for evaluating the students' understanding of the material, and recommendations for the use of additional assignments.

A. PRESENTATION OF THE COURSE

To use the *VideoSoft* Instructional Series the minimum equipment needed is a video cassette player/recorder (VCR) and a television monitor. The teacher will find search, memory and pause options to be useful additions to the VCR if portions of a lesson are to be replayed.

While student access to computers is not absolutely necessary, student computer labs are recommended.

1. Use of Video Tapes

The *VideoSoft* Instructional Series consists of fourteen lessons, each on a separate video tape. Most lessons are from forty to fifty minutes in length. The running time for each lesson is noted at the beginning of the teacher's notes for each lesson in the Teacher's Edition.

All video tapes begin with color bars followed by thirty seconds of blank tape. The *VideoSoft* logo is then displayed for ten seconds followed by the lesson card for seven seconds.

The first lesson tape varies from the other lessons in that the lesson card follows the instructor's introduction to the course.

Each video tape ends with twenty to thirty seconds of the command board immediately following the assignment. The command board is a large board on which the BASIC and DOS commands are displayed as they are learned.

The video tape should be advanced to the *VideoSoft* logo and played from that point. The seventeen seconds of logo and lesson card serve as a transition allowing the students to mentally prepare for the beginning of the instruction.

The answers to the assignments are included on two separate tapes. The answers to lessons one through eight have been placed on Answer Tape One. The answers to lessons nine through thirteen have been placed on Answer Tape Two. Lesson fourteen has no assignment.

Each answer begins with seven seconds on the lesson card. Lesson answers are separated on the tape by ten seconds of blank tape.

2. Presentation Of The Lessons

The design of the *VideoSoft* Instructional Series allows its use in virtually any educational environment. The video tapes can serve as the core of the programming instruction, or as a tool to enhance and clarify programming concepts in conjunction with other instruction.

While the use of the series is limited only by the creativity of the teacher, the course fits conveniently into four educational environments: 1) an eighty to ninety day school semester, 2) a full 175 to 180 day school year, 3) a thirty day summer semester, or 4) the intensive three or four week computer camp.

a. School Semester

The *VideoSoft* Instructional Series is suited to serve as the basis for an eighteen week, one semester course in BASIC computer programming.

A typical structure for each lesson might be as follows:

1. View the lesson tape and assign the Self-Check Tests for homework.
2. Review the lesson by discussing the Self-Check Tests and more thoroughly explaining the programming assignment. At this time, the teacher may wish to replay portions of the tape which were in any way unclear to the student. The student should now be prepared to answer the assignment.
3. Each student should be given a supervised lab day. At this time the student should enter and debug his or her program on a computer.
4. View the appropriate lesson answer tape and discuss the course's answer and the students' answers. Assign one or more additional assignments.
5. Review the answers to the additional assignments. The answers to the additional assignments will serve as a final review of the lesson.

Each Evaluation Test should be preceded with a review of the material covered. The test should then be given, followed by an explanation of the answers to the test.

If the above schedule is followed uninterrupted, the *VideoSoft* series will require approximately 75 class days. The teacher will therefore have the flexibility of utilizing additional class periods for the more complex concepts.

b. Full School Year

When providing a two semester course in BASIC computer programming, the *VideoSoft* Instructional Series may be used in conjunction with a course in computer literacy, used as the introductory programming course, or used as an additional resource for the programming course.

If the *VideoSoft* Instructional Series is to be used with a course in computer literacy, the literacy course should be taught in the first semester. The *VideoSoft* BASIC programming course should be taught in the second semester. Using this approach, the *VideoSoft* course will provide meaning and substance to the abstract concepts covered in most literacy courses. In this format, The *VideoSoft* course would follow the recommendations outlined for the one semester course.

If a two semester course in BASIC programming is to be taught, the *VideoSoft* series can serve either as the first semester of the course, or as a resource to provide clarification of the programming concepts and as a source of additional programming assignments.

The *VideoSoft* Instructional Series will provide the student with a solid base in BASIC programming. The teacher's efforts and creativity can then be addressed to the development of the second semester. The second semester should focus on teaching the student how to use the BASIC programming commands to solve specific programming tasks. The second semester should also cover the advanced programming

commands not covered in the initial *VideoSoft* series, such as the higher math functions and user-defined functions.

The *VideoSoft* Instructional Series can be easily interfaced with an existing BASIC programming course. The teacher will find the *VideoSoft* course to be an invaluable tool when explaining the concepts and commands of the BASIC programming language. The teacher will also find the lesson assignments and additional assignments very useful when creating and explaining programming exercises which utilize the commands and skills learned by the students.

c. Summer Semester

Use of the *VideoSoft* Instructional Series in a summer semester assumes a six week, 30 day semester with a minimum of one hour class time. To enter and debug programs, computer lab time should be allocated in addition to the class time.

The class days should alternate between presentation of the lesson on one day and explanation of the answer to the assignment and discussion of the Self-Check Tests on the next. The answer to the assignment and discussion of the Self-Check Tests serve as a review of the previous lesson. When using the alternating day format, the course will require approximately twenty-seven days. The Evaluation Tests should be worked into the class schedule as appropriate.

d. Computer Camp

When using the *VideoSoft* Instructional Series in a computer camp, the following general format is recommended.

Each day should begin with the answer to the previous lesson's assignment followed by a discussion of the course's answer and the students' answers. Discussion of the assignment answers serves as a review of the commands and concepts covered in the previous lesson.

The next lesson should then be viewed followed by a more thorough explanation of the assignment for that lesson. The explanation of the assignment serves as a review of the lesson and will uncover commands or concepts which are in any way unclear.

To further clarify commands and concepts, the teacher should expand on the command or concept and/or replay the appropriate portion of the lesson.

The student should then prepare, enter and debug his or her answer to the assignment on a computer.

3. Use Of The Student Study Guide

The Student Study Guide is designed to compliment and extend the video taped lessons. Proper use of the Study Guide can enhance the learning process and improve retention.

Prior to viewing a video taped lesson, the student should prepare by reading the appropriate section in the Study Guide. The student should first read the lesson objectives and look up the explanation of each command and concept in the Glossary and Summary of Commands in the back of the guide. The student should then scan the programs to be used in the lesson. Previewing the lesson commands and concepts allows the student to focus on the explanation, clarification and examples used in the lesson.

After viewing the lesson, the student will find the explanation of the assignment in the Study Guide to be especially helpful as the assignment given in the lesson is often abbreviated, particularly in the later lessons.

The student will find the Glossary, Summary of Commands and Reserved Word List in the back of the Study Guide to be very useful for quick reviews of the commands and concepts.

B. EVALUATION OF STUDENT PROGRESS

The *VideoSoft* Instructional Series provides Self-Check Tests with each lesson and Evaluation Tests

covering logical groupings of lessons. Both tests provide the teacher with a means of evaluating the students' understanding and retention of the concepts and commands. The Self-Check Tests cover the lesson material in detail whereas the Evaluation Tests concentrate on the most important concepts.

Since each lesson builds on the previous lessons, it is important that any unclear concepts or commands be identified and clarified promptly. Assigning and reviewing the Self-Check Tests after each lesson will assist the teacher in identifying points of weakness in the students' understanding.

The Evaluation Tests provide the teacher with a means of periodically evaluating the students' comprehension and retention of the important concepts and commands. The teacher may review the lessons covered in an Evaluation Test prior to administering the test. A second class period should be allocated to a discussion of the Evaluation Test answers.

C. ENHANCING AND EXPANDING STUDENT UNDERSTANDING

The *VideoSoft* Instructional Series provides two methods of enhancing and expanding the students' understanding of the concepts and commands necessary to program a computer in BASIC: (1) additional assignments, and (2) review of selected portions of the lesson tape.

1. Using Additional Assignments

Each lesson, except the last, includes two additional assignments: one of the additional assignments provides practice and review of the concepts and commands. The second assignment requires the student to use the material more creatively.

The single most important factor when learning to program a computer is to learn to write and debug programs. For this reason, the teacher is encouraged to utilize all additional assignments.

The ideal learning technique is for the student to answer additional assignments on paper and then enter and debug the program on a computer. If computer lab time is not available, the course answers should be explained and discussed with the class, and the teacher should evaluate each student's program. To simplify the teacher's evaluation, all assignments are very specific and answers are provided in the back of the Teacher's Edition.

Replaying Video Lessons

The teacher will find replaying all or portions of the *VideoSoft* Instructional Series to be particularly valuable. If the student has unsuccessfully attempted to answer an assignment, reviewing the video taped lesson will often make the concept or command very clear.

The teacher is encouraged to schedule lesson replays for students whose understanding is in any way lacking.

TEACHING SUGGESTIONS

The Teaching Suggestions include both general suggestions to the teacher and lesson-by-lesson teaching suggestions and notes.

A. GENERAL TEACHING SUGGESTIONS

1. Instructional Preparation

The teacher should become familiar with the VideoSoft course prior to using the video taped lectures in the classroom. If time permits, it is recommended that the teacher take the VideoSoft BASIC programming course before attempting to teach it.

2. The "Correct" Answer

No two computer programmers will approach a task in exactly the same way. While all assignments have been written with specific requirements, the teacher will find that each student's approach may be slightly different; some will be drastically different. As long as the student's program includes the specific requirements of the assignment, the student's answer should be considered correct. In fact, the student should be encouraged to approach assignment answers creatively since a creative answer further exercises the student's knowledge and understanding of the commands and concepts.

In other words, there is no one correct answer. There are only answers which meet or fail to meet the requirements of the assignment.

3. Modular Programming

Modular programming is the technique of writing computer programs in program sections each of which executes a specific portion of the programming task. The VideoSoft Instructional Series follows modular program development throughout the course.

Modular programming encourages the student to think of a programming task as a set of specific steps which can be entered and debugged one at a time. This modular logical process is directly transferable to programming in more structured languages and applies equally well to non-computer problem solving.

4. Line Numbering Protocol

Throughout the VideoSoft Instructional Series, program line numbers are used not only to control program flow, but also to superimpose programming structure on an unstructured programming language, BASIC. "Line number protocol" simply refers to the use of different ranges of line numbers for different purposes.

When examining additional source materials, the teacher will find many different line numbering protocols and will find many resource books follow no line numbering protocol whatsoever.

The VideoSoft line numbering protocol facilitates the student's eventual transition to the more structured programming languages such as PASCAL or COBOL and encourages modular programming resulting in programs which are easier to write, easier to debug and more reliable.

5. Additional Resource Materials

The teacher will find an abundant quantity of resource books available on the concepts of BASIC programming. The use of additional resources is at the discretion of the teacher. However, the teacher is cautioned to enter and RUN programs from other sources to make sure they are correct and workable.

All programs in the VideoSoft Series have been entered and RUN by at least three different programmers to ensure workable, bug-free programs. Unfortunately, many programs listed in resource books

and magazines contain "bugs". Incorrect model programs are frustrating and detrimental to the student's learning.

The teacher should also examine all additional resource material to verify that the programmer utilized some form of program structure. If the student is presented with an obviously unstructured program, the student will receive cross-messages and may be led to consider a lack of structure as an acceptable programming technique.

6. Error Messages

When the computer encounters errors during program execution, an "ERROR MESSAGE" is issued along with the line number in which the error was encountered. The most common error messages are explained where appropriate in the lessons. The student, when encountering an error message, should list the line in which the error was found and the error will usually be apparent.

B. LESSON NOTES

In this section of the Teacher's Guide, teaching suggestions and notes are provided where applicable on a lesson-by-lesson basis.

The specific lesson objectives and list of new commands are found at the beginning of each lesson in the *Student Study Guide* and are not repeated in the Lesson Notes.

The teacher should first read the lesson objectives described at the beginning of each lesson in the *Student Study Guide* before reading the Lesson Notes.

C. LESSON EXTENSION EXPLANATIONS

Where appropriate, information which may be used to extend course concepts is provided on a lesson-by-lesson basis. The lesson extension is placed at the end of the normal lesson notes.

The lesson extensions are not necessary to the course, but may be helpful if the teacher wishes to extend the concepts covered.

Lesson One: Introduction
Viewing Time: 40:06

Lesson One serves as an introduction to the course and introduces the student to the first BASIC programming commands.

The BASIC language was developed by two Dartmouth College mathematicians, Dr. John Kemeny and Dr. Thomas Kurtz, in 1965.

The simulated keyboard will be helpful in this lesson to assist students in learning the computer keyboard.

The initial commands were selected to move the student quickly into programming.

Lesson Two: Basic Printing Commands and DOS Commands
Viewing Time: 47:49

In this lesson the student is introduced to the first of the VideoSoft line numbering protocols: the main body of the program begins at line 1000. Line 10 includes a REMark briefly describing the program. Line 20 includes the program SAVE command in a REMark.

If a student permanently RENAMEs a BASIC program file, the student should be reminded to change the SAVE command listed in the REMark in line 20.

EXTENSION: In Applesoft BASIC the listing of files on a diskette is called a "CATALOG". The Catalog is known as a "VTOC" (Volume Table Of Contents) on most larger machines. The file listing a diskette is called a "DIRECTORY" on some other small computers.

Lesson Three: Variables
Viewing Time: 37:34

Lesson Three introduces the student to the concept of BASIC variables.

A variable is a place in the memory of the computer in which information is stored. The VideoSoft lesson uses the analogy of a box to represent a variable.

The VideoSoft Instructional Series usually uses two-letter variable names throughout the course because the computer only recognizes the first two letters of any variable. However, many programmers prefer to use longer, specific variable names such as "AMOUNT" instead of "AM". Use of longer variable names is often called "COBOL documentation" because descriptive variable names originated in the COBOL programming language. Earlier versions of BASIC did not allow long variable names because the creation of BASIC was heavily influenced by FORTRAN, which uses simple variable names.

The VideoSoft Instructional Series uses the terms "PROGRAMMER" to refer to the person who writes and debugs programs, and "OPERATOR" to refer to the person who RUNs the program. While the term "PROGRAMMER" is fairly standard, the "OPERATOR" is often referred to as the "USER".

The terms "numbers only" and "character" variables are used for descriptive purposes only. The technically correct names are "numeric" and "string".

Lesson Four: Math & Formatted Printing
Lesson Viewing Time: 54:38
Math Viewing Time: 25:16
Formatted Printing Viewing Time: 28:50

In Lesson Four the student is taught the fundamental calculating capabilities of the computer and the commands used to format the CRT (Cathode Ray Tube).

Lesson Four may be taught in two separate classes. Two assignments, one following the math portion of the lesson and one following the formatted printing portion, are given to facilitate this.

In this lesson the student is introduced to another line number protocol. String and numeric literals are assigned to variables prior to the beginning of the main program in line 1000. These initial program operations will eventually be called the "Housekeeping Section" and utilize the line numbers smaller than 1000.

An explanation of Scientific Notation (E-Notation) is included in the back of the Study Guide.

If your computer model has an 80 column screen instead of a 40 column screen, the same formatting operations apply, except each screen line has 80 printing positions instead of 40.

EXTENSION: The print formatting capabilities of the two punctuation marks, the comma and semicolon, apply ONLY when used following the PRINT command. The comma and semicolon are also used to separate different parts of commands. For example, the semicolon is used in a prompted input to separate the prompt from the variable. When the semicolon and comma are used in this form, they are called "delimiters".

Students with an advanced math background may be interested in the higher math functions of BASIC. A listing and description of the higher math functions is included in the manuals which come with the computer under the heading "FUNCTIONS".

Lesson Five: Decisions
Viewing Time: 39:50

In Lesson Five the student is introduced to the decision making capabilities of the computer through the various uses of the IF-THEN command pair.

In this lesson the student first encounters ending routines for programs beginning at line 9000, another line numbering protocol. While Applesoft allows five-digit line numbers, only four-digit line numbers are used in this course because many versions of BASIC limit line numbers to four digits.

In this lesson the instructor uses the term "GOTO LOOP". This is not a technical term, but is rather used to familiarize the student with the term LOOP and the concept of a LOOP. This lays the groundwork for later lessons.

The computer you use may allow RESET without the control key.

Lesson Six: Programming Techniques
Viewing Time: 47:41

In this lesson the student is introduced to numerous techniques which may be used in Applesoft BASIC when writing programs.

The question mark for the PRINT command and the optional use of the semicolon in a PRINT line are not standard BASIC but are allowable in Applesoft and may be allowable in some other versions of BASIC.

The student should understand that while the semicolon is optional under certain conditions in a PRINT line, it is required syntax in a prompted INPUT.

Furthermore, the student will find the semicolon sometimes necessary for clarity. For example, if the student attempts to print two variables next to each other on the same line without a semicolon, the computer may not recognize them as separate variables.

In the "NUMBER-GUESS" program the student encounters another VideoSoft line number protocol: in longer programs each section (or module) begins on an even 100 or 1000 line number.

The VideoSoft instructor will recommend that the student follow along in the Study Guide as he details the Number Guess Outline and logic flow.

Lesson Seven: Loops
Viewing Time: 49:40

In Lesson Seven the student is introduced to the "FOR-NEXT" command pair and the "STEP" command to define the limits of a loop.

In this course accumulator variables are used only to count the number of times a series of programming lines are repeated. Variables may also be used as accumulators when accumulating the total of a column of figures in accounting programs. They are therefore sometimes called "TOTALLING" variables.

While any numeric variable may be used as a loop variable, the variables "I" and "J" are traditionally used. The use of "I" and "J" reflect the influence of FORTRAN on BASIC. In FORTRAN variable names beginning with "I" and "J" are assumed to be integers, or whole numbers, which the computer can execute more rapidly since no decimal point needs to be maintained. Therefore, FORTRAN loops normally use the integer variable names "I" and "J" as loop variables to improve execution speed.

EXTENSION: The VideoSoft BASIC course teaches using a positive whole step (STEP 1) and a negative whole step (STEP -1). An interesting side note to the STEP command is that the step need not be a whole number. In fact, in scientific applications, it is not uncommon to find a FOR-NEXT loop with a "STEP .1" or a "STEP .05".

Lesson Eight: Subroutines
Viewing Time: 41:04

In Lesson Eight the student is introduced to subroutines, branching and flowcharting.

Using the VideoSoft line numbering protocol, subroutines are placed following the main program module usually beginning on an even 1000 line number or an even 100 line number.

The VideoSoft Instructional Series introduces the student to both the flow chart method of program design and to the outline/logic paragraph method of program design (in Lesson Six). The value of flowcharting is the subject of considerable controversy among data processing professionals. Some experts describe flowcharting as invaluable while others consider flowcharting to be practically worthless as a design tool. Most professional programmers lean toward the outline/logic paragraph technique when designing the specific program steps. Flowcharting is used extensively for Systems Design. The program next to the flowchart is presented in the Study Guide to illustrate a program that could have been written from the chart. The program is not run in the lecture.

EXTENSION: Where execution time is critical, programmers place frequently used subroutines near the beginning of the program and jump over the subroutines with a GOTO command at the beginning of the program. The improved execution time results from the procedure used by Applesoft to find a subroutine. The computer begins searching with the first line number of the program and continues until the beginning line number of the subroutine is found. Therefore, subroutines at the beginning of the program execute a little more rapidly.

Lesson Nine: Arrays
Viewing Time: 49:45

In Lesson Nine the student is introduced to the concept of arrays.

In this lesson the student is introduced to the last of the VideoSoft line numbering protocols: the DIMension statements for arrays is placed at the very beginning of the program in lines 50 through 90 and are a part of the program's "Housekeeping Section".

In some computers the zero element of an array is not allowed. In other computers, the programmer may define the arrays in a given program as base zero or as base one.

Many professional programmers do not use the zero element of arrays because use of the zero element makes the program more difficult to write and read. It should be acceptable for the student to dimension an array using either a zero subscript or a one subscript as the first element.

EXTENSION: An array, especially a two-dimensional array, is frequently called a "Matrix", and many versions of BASIC have special commands, MAT commands, which allow the programmer to perform an operation with an entire array. A one-dimensional array is sometimes called a "Vector".

Many of the earlier versions of BASIC allowed only numeric arrays because BASIC arrays follow the same general format as FORTRAN arrays. Most versions of FORTRAN allow only numeric arrays.

Lesson Ten: String Functions
Viewing Time: 44:53

In Lesson Ten the student is introduced to the computer's string functions.

In this course the string functions are used primarily to format CRT displays. However, in more advanced programming, the string functions are used in numerous applications, particularly in advanced uses of data files.

The plus sign "+", when used with numbers or numeric variables adds the values in a math operation. Concatenation is only possible with string literals or string variables.

When a string has no value, it is called a "Null String". A null string has a length of zero. For example, the statement A$="" creates a null string, and in the statement L=LEN(A$), the variable "L" would equal zero. The null string assignment is used to initialize the value of a string variable. The following two statement lines could be used to determine if the operator pressed the "RETURN" key in response to an INPUT:

```
1000 INPUT "(PRESS RETURN TO END) ";A$
1010 IF A$="" THEN END
```

Note: In line 2020 of the "LEN-FUNCTION" program, the optional semicolon was not used between the variable "W$" and the string literal.

Before assigning the second additional assignment, the student should have viewed the course answer to the lesson assignment. The student will also find a listing of the answer to the lesson assignment to be helpful when attempting to answer the second additional assignment.

NOTE: IF THE CLASS IS RUNNING SHORT ON TIME, LESSON 12 OR LESSONS 11 & 12 MAY BE OMITTED WITHOUT ANY SERIOUS LOSS TO THE STUDENT.

HOWEVER, WE FEEL BOTH YOU AND YOUR STUDENTS WILL FIND THE LESSONS ON GRAPHICS AND ANIMATION BOTH EDUCATIONAL AND ENJOYABLE.

Lesson 11: Lo Res Graphics
Viewing Time: 44:16

In Lesson Eleven the student is introduced to the lo resolution color graphics commands of the computer.

Near the beginning of the tape, a program is run which displays the colors along with names and then the colors along with the numbers. The teacher may wish to adjust the "TINT" control on the monitor at this time so the colors used in the lecture will reproduce correctly.

If the student is following the lecture on an individual computer, you may wish to pause the tape and allow the student to adjust the CRT.

The tape picture will "jump" slightly each time the computer switches from black & white to color or from color to black & white. The "jump" is a result of the Apple's video signal and not a malfunction of tape or equipment.

Lesson 12: Animation
Lesson Viewing Time: 58:05
Viewing Time 1st Half: 31:58
Viewing Time 2nd Half: 26:07

In Lesson Twelve the student is introduced to the concepts of graphic animation. While teaching the student to move a block of color across the screen, the student is taught how to directly access the contents of the computer's memory locations.

Due to the length of viewing time, Lesson Twelve may be viewed in two parts. The teacher will find the most convenient break after the "ROAMING BLOCK" program and just prior to the introduction to the PEEK and POKE commands.

STUDENT STUDY GUIDE

VideoSoft™ Instructional Programming
Series for Applesoft™ Basic

Copyright 1983 by *VideoSoft*, Inc.
All rights reserved. With the exception of the Evaluation Tests which may be reproduced for testing purposes, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

Manufactured in the United States of America

®APPLE is a registered trademark of Apple Computer Inc.

VideoSoft™ Instructional Programming Series For Applesoft™ BASIC

TABLE OF CONTENTS

I. Course Outline 1
II. Lesson One 4
III. Lesson Two 10
IV. Lesson Three 17
V. Lesson Four 21
VI. Lesson Five 29
VII. Lesson Six 35
VIII. Lesson Seven 43
IX. Lesson Eight 51
X. Lesson Nine 60
XI. Lesson Ten 68
XII. Lesson Eleven 77
XIII. Lesson Twelve 86
XIV. Lesson Thirteen 98
XV. Lesson Fourteen 106
XVI. Appendices
1. Answers to Self-Check Tests 117
2. Scientific Notation 121
3. Glossary of Terms 122
4. Summary of Commands 127
5. Reserved Words 131
6. Color Number Chart 131
7. Lo-Resolution Screen Grid 132

VIDEOSOFT COURSE OUTLINE

LESSON ONE — INTRODUCTION

Introduction to the course. Components of the lessons. History of the computer. Introduction to BASIC. Introduction to the computer: how to turn the computer on, the keyboard. Beginning to program: Diskette organization, initialization. BASIC and DOS commands. Writing a greeting program. **NEW. HOME. PRINT.** Statement line. Adding a line out of order. Using direct commands. **RUN. LIST.** Order in which the computer processes a program. Greeting program and initializing a diskette. **INIT.** Summary. Assignment.

LESSON TWO — BASIC PRINTING COMMANDS AND DOS COMMANDS

Brief review of Lesson One. Introduction to the lesson. List of new commands. Review of **PRINT** command. Beginning main section on line 1000. **INVERSE. NORMAL. FLASH.** Deleting a BASIC statement line. Inserting a new line number. Numbering lines in increments of 10. **SPEED. SPEED = 0.** Different printing rates. Combining the **PRINT** modes. **END. PRINT** command to print a blank line. String literal. Review of commands and suggestions.

More DOS and BASIC commands. **SAVE. REM.** Rules for naming files on the Apple Computer. **CATALOG. LOAD. RUN** as a DOS command. **RENAME. DELETE. LOCK. UNLOCK.** Listing a portion of the program.Summary. Assignment.

LESSON THREE — VARIABLES

Definition of a variable. Naming the variable. Numeric variables. Character variables. Rules for naming variables. Assigning values to variables. **LET. INPUT.** Differences in string (character) and numeric variables. Using the **PRINT** command to display variable values. Forming a character string. Assumed **LET**. Correct and incorrect use of variables. Restrictions on variable names. Reserved words. Review. Reassignment of variable values. **INPUT**. The Prompt. Combining **PRINT** and **INPUT (PROMPTED INPUT).** Structure of Prompted Input using character and numeric variables. Adding remarks to program. Summary. Assignment.

LESSON FOUR — MATH AND FORMAT PRINTING

Introduction. Five symbols for math operations: +, −, *, /, ^. Exponentiation. Storing math calculations in a variable. Performing the five math functions with the computer. Allowing **INPUT** from operator for calculations. Multiple calculations. Order in which they are performed. Changing priorities. Display of numbers. E-Notation. Review of Math. Assignment for Math.

Definition of screen format. Introduction to screen formatting section. Formatting techniques. Text Screen. Wrap around using long print line. Location of the home position. Using commas and semicolons. Commas and print zones. Using semicolons to combine numeric variables, character variables and literals. Formatting commands: **VTAB. TAB ( ). HTAB.** Differences in **TAB( )** and **HTAB.** Summary of Formatted Printing. Assignment for Formatted Printing.

LESSON FIVE — CONDITIONING COMMANDS.

Brief review of previous lessons. Introduction. **GOTO.** Control-C and Control-Reset. Eternal loop (or **GOTO** loop). **IF-THEN. IF-THEN** with **GOTO. IF-THEN** with **PRINT.** Comparisons using = , > , < , and <> Establishing multiple conditions with **AND** and **OR.** Testing for =, >, <, and <> with **IF-THEN**. Summary. Assignment.

LESSON SIX — PROGRAMMING TECHNIQUES, PART ONE

Introduction to the lesson. The question mark (?) as shorthand for **PRINT**. Dummy Input. Multiple Statement lines. Optional semicolon when printing variable and string literals on the same line. Multiple **INPUT** Statements. **CHR$ (7).** Control-S key to scan. Deleting multiple program lines with **DEL** Review of programming

techniques. Program Design. Outlining and writing a long program. Modules or Sections. Program outline and logic paragraphs. Analysis of "NUMBER-GUESS" program: Housekeeping Section, Instructions to Players and Title Page, Rules of Game, Play of Game, Ending Section. Summary. Assignment.

LESSON SEVEN — LOOPS

Introduction. Accumulator. Loops. Use of the **IF-THEN**. Combining functions of the accumulator and the **IF-THEN. FOR-NEXT** loops. Timing Loops. Using variables for the beginning and ending values of a **FOR-NEXT** loop. **FOR-NEXT** with **STEP. STEP**ping by numbers other than one. **STEP**ping backward. Nested Loops. Example of Nested Loop. Summary. Assignment.

LESSON EIGHT — SUBROUTINES

Introduction. The paired commands **GOSUB** and **RETURN**. Structure of a Subroutine. Common errors. **ON ... GOSUB. ON ... GOTO.** Program design using Flowcharting. Flowcharting Template. Five of the flowcharting symbols: Terminal, I/O Operation, Decision, Process Block, Predefined Process. How to flow a program. Summary. Assignment.

LESSON NINE — ARRAYS

Definition of Arrays. Elements. Arrays as subscripted variables. Subscripting on a zero based machine. Dimensioning an array. **DIM**. Dimensioning more than one array on the same line. Placing **DIM** statements in housekeeping section. Using a variable as a subscript number. A comma in the operator's response to an **INPUT** statement. **READ** and **DATA**. Making a program "user friendly". Multi-dimensional arrays. Dimensioning a Multi-dimensional array. Referencing an element in a multi-dimensional array. Summary. Assignment.

LESSON TEN — STRINGS

Manipulating character strings for more sophisticated screen formatting. String manipulation functions. **LEN ( ). LEFT$ ( ). RIGHT$ ( ). MID$ ( ).** Putting a date into a readable form. Extracting characters from a string. Using the plus "+" for Concatenation. **VAL ( ). STR$ ( ).** Review. Extracting characters one at a time. Null string. Summary. Assignment.

LESSON ELEVEN — LO RES GRAPHICS

Introduction. LO RES Graphics. Using the "COLOR DEMOSOFT" program: Color names, Color numbers. **GR.** The graphics screen. The Text Window. Numbers of columns and rows. **COLOR=. PLOT. TEXT. HLIN. VLIN. SCRN ( ).** Drawing pictures in LO RES Graphics. How to change an existing color. Summary. Assignment.

LESSON TWELVE — ANIMATION

Introduction. Creating animation by drawing and erasing. Erasing by redrawing the background color. Creating fluid motion. Review. Random movement. **RND (1).** Structure of **RND.** Decimal numbers. Selecting one of four directions. Removing the decimal portion of a number. **INT ( ).** Allowing the operator to control animated movement. Another way to enter information from the keyboard. **PEEK ( ). POKE.** Memory addresses. **PEEK**ing at the keyboard. Using the keys to enter information. Using **PEEK ( )** and **POKE**. Modifying the BELL sound. Creating illusion of movement over multicolored background. Summary. Assignment.

LESSON THIRTEEN — DATA FILES

Introduction. Data Files. Storing and retrieving data. Structure of the DOS commands used with Data Files. Sequential Files. **CLOSE. OPEN. WRITE. PRINT** as a DOS command. **READ. INPUT** as a DOS command. Storing data in a sequential file with the **WRITE** and **PRINT** commands. "T" in the **CATALOG** listing. Preparing the file for use with the **OPEN** command. Retrieving information from a sequential file with the **READ** and **INPUT** commands. Adding data to a sequential file. **APPEND**. File Pointer. Random Access Files. Storing data in a random file. Retrieving data from a

random file. **ONERR GOTO.** Assigning record numbers using a Control Record. Summary. Assignment.

LESSON FOURTEEN — PROGRAMMING TECHNIQUES, PART 2

Introduction. Useful tips and techniques. Modifying programs written by others. **GET.** Scrolling Window. Another kind of **IF-THEN. RESTORE.** Machine Language Subroutine. **CALL.** Tools to increase programming speed and effectiveness. Using DOS commands in a BASIC program. Editing techniques. Entering Edit mode. Escape key and "I", "J", "K", or "M". How to edit a line. Other Escape key operations: "E", "F", "@". Control—X. **TRACE. NOTRACE.** Computers as a logic tool. Conclusion of course.

LESSON ONE: *INTRODUCTION*

LESSON OBJECTIVES

Lesson One includes an introduction to the course, a brief history of computers, a brief introduction to the computer which will be used during this course, and the first BASIC programming commands.

You will learn how to enter a program, how to display a program, how to execute a program, and how to prepare a diskette for use.

The commands introduced in this lesson are: **PRINT, HOME, NEW, LIST, RUN** and **INIT**.

Program Listing For Lesson 1

Greeting Program

```
10  HOME
20  PRINT "HELLO"
30  PRINT "VIDEOSOFT DISKETTE"
40  PRINT "BEN"
```

Lesson One Self-Check Test #1

Directions — You are to match the answer at the bottom of the page with the correct definition or statement by writing the letter of the correct answer in the space provided. Please note there are more choices than definitions or statements. The answers can be used only once.

——— 1. Message given by the computer for improper entry.

——— 2. White block of light that shows placement on the screen.

——— 3. The T.V. screen for the computer.

——— 4. Instructions from a program go to this chip, which in turn determines what all the other chips do.

——— 5. Gives instructions to the computer with words or groups of words.

——— 6. Gives instructions to the computer with letters.

——— 7. Gives instructions to the computer with numbers.

——— 8. Errors in a computer or program.

——— 9. Used vacuum tubes instead of mechanical relays.

——— 10. Programming was done with a wrench and a screwdriver.

——— 11. With cards that had punched holes, and a machine he found a way to do the 890 census in two and one-half years.

——— 12. Known as the "Father of the Computer".

——— 13. Earliest known device to perform calculations for mankind.

A . ABACUS
B . ASSEMBLY LANGUAGE
C . CHARLES BABBAGE
D . BASIC
E . BUGS
F . CRT (CATHODE RAY TUBE)
G . CURSOR
H . DIFFERENTIAL ANALYSER
I . ENIAC
J . HIGH LEVEL LANGUAGE
K . HERMAN HOLLERITH
L . MACHINE LANGUAGE
M . MARK I
N . PROCESSOR
O . SYNTAX ERROR

Lesson One Self-Check Test #2

Directions — Match the items from the key with the statements below. Write the letter in the space or spaces provided. Items may be used more than once.

A . CTRL	E . →
B . KEYBOARD	F . ←
C . ESC	G . RETURN
D . REPT	H . SHIFT

——— 1. Used to give commands to the computer.

——— 2. Used with keys that have two symbols.

——— 3. Used with another key like "RESET" or the "C" key.

——— 4. Used to move the cursor to the left.

——— 5. Used to process (execute) a command.

——— 6. Used more often in advanced programming.

——— 7. Used with computer's panic buttons.

——— 8. Used to print a series of the same character.

——— 9. Used when you do not wish to erase what is written while moving the cursor.

——— 10. Used to move the cursor to the right.

Lesson One Self-Check Test #3

Directions — If the statement is TRUE, write a "T" in the space provided. If the statement is FALSE, write an "F" in the space and change the italicized portion to make the statement TRUE.

——— 1. In a print command line the ***commas*** tell the computer what to display.

——— 2. The number zero and the letter "O"' are ***interchangeable***.

——— 3. Computer programming is giving the computer a set of ***instructions*** to follow.

——— 4. If the programs are not stored on ***diskette*** they are lost when the computer is turned off.

——— 5. ***DOS*** is the language used to control files, programs, or data on diskette.

——— 6. Diskettes in the drive are used for ***temporary*** storage of programs and data.

——— 7. ***Initialization*** is the process of dividing a diskette into tracks and sectors.

——— 8. A diskette (which is actually a flat piece of recording tape) is sometimes called a ***droopy***.

——— 9. The computer processes a program by reading instructions in order from the ***lowest*** program line to the ***highest***.

——— 10. The PRINT ***command*** preceded by a line number does not immediately print.

Lesson One Self-Check Test #4

Directions - The following are definitions of some of the commands and terms used in Lesson One. Write the correct command or term in the space provided.

_____ 1. High level language which this course will be using to write programs for the computer.

_____ 2. Command used to initialize the diskette, followed by the name of the program.

_____ 3. Tells the computer to display a program stored in memory.

_____ 4. What a command is called when it is given to the computer without a line number.

_____ 5. Command that clears the screen (CRT).

_____ 6. Command that clears the memory.

_____ 7. Command that instructs the computer to display something on the CRT.

_____ 8. Recording tape affixed to a solid core.

_____ 9. Instructs the computer to process the program it currently has in memory.

_____ 10. What the PRINT command followed by words in quotation marks is called.

_____ 11. Program required by Applesoft to initialize a diskette.

_____ 12. Word used to describe the process of inserting a diskette into the drive and then turning on the computer.

Assignment For Lesson One

Your assignment for Lesson One is to write a "Greeting Program."

The program should do the following:

1. Clear the screen using the HOME command.
2. Use the PRINT command to display the name of the "Greeting Program" on the CRT.
3. Use the PRINT command to identify the diskette.
4. Use the PRINT command to display your name.
5. Use the "Greeting Program" and the INIT command to initialize a diskette.

Additional Assignment For Lesson One, #1

Write a program which prints the following on the CRT:

"THE HOME COMMAND"
"CLEARS THE SCREEN."

"THE PRINT COMMAND"
"DISPLAYS MESSAGES"
"ON THE SCREEN."

Be sure to clear the screen in the first line of the program.

Additional Assignment For Lesson One, #2

Write a program which clears the CRT and prints the following:

```
XX     XX
 XX  XX
   XXX
 XX  XX
XX     XX
```

To print the X's, remember that blanks enclosed within quotation marks are characters.

LESSON TWO: *BASIC PRINTING COMMANDS AND DOS COMMANDS*

LESSON OBJECTIVES

In Lesson Two you will learn some additional BASIC commands which can be used in conjunction with the **PRINT** command. You will also learn how to use the DOS *(Disk Operating System)* commands.

The first portion of the lesson introduces the BASIC commands which may be used with the **PRINT** command to create a more interesting and attractive display.

You will learn how the computer uses *line numbers* to control the order in which commands are executed. You will also learn the definition of a *string literal*, any combination of characters enclosed within quotation marks.

The new BASIC commands introduced in this lesson are: **INVERSE, NORMAL, FLASH, SPEED, END** and **REM.**

The second portion of the lesson explains the use of the DOS *(Disk Operating System)* commands.

DOS is a specialized, high-level language limited to controlling the diskettes. You will learn that diskettes are divided into files and that BASIC programs are stored in files on diskette.

The new DOS commands introduced in this lesson are: **SAVE, CATALOG, LOAD, RUN** as a DOS command, **RENAME, DELETE, LOCK** and **UNLOCK.**

Program Listing For Lesson 2

Normal/Inverse/Flash Demonstration

```
1000  HOME
1010  PRINT "NORMAL PRINT MODE"
1020  INVERSE
1030  PRINT "INVERSE PRINT MODE"
1040  NORMAL
1050  PRINT "BACK TO NORMAL"
1060  FLASH
1070  PRINT "FLASHING MODE"
1080  NORMAL
1090  PRINT "BACK TO NORMAL"
```

Speed Program

```
1000  HOME
1010  SPEED= 255
1020  PRINT "SPEED 255 IS NORMAL"
1030  SPEED= 64
1040  PRINT "SPEED 64"
1050  PRINT "PRINTS THIS SLOWLY"
1060  SPEED= 0
1070  PRINT "0(ZERO)-THE SLOWEST SPEED"
1080  PRINT "PRINTS LIKE THIS"
1090  SPEED= 255
1100  PRINT "BACK TO NORMAL SPEED"
```

Print Modes Program

```
10  REM :: PGM TO SHOW PRINT MODES
20  REM  SAVE PRINT MODES
1000  HOME
1010  SPEED= 0
1020  INVERSE
1030  PRINT "S - T - E - P"
1040  PRINT "  BY   STEP  "
1050  SPEED= 255
1060  FLASH
1070  PRINT
1080  PRINT "%?$ CRASH !!! %?$"
1090  PRINT
1100  SPEED= 20
1110  PRINT "M E O W !!!!!!"
1120  NORMAL
1130  SPEED= 255
1140  END
```

DOS Experiment Program

```
10  REM :: DOS EXPERIMENT
20  REM  SAVE DOS EXPERIMENT
1000  INVERSE
1010  PRINT " A DOS"
1020  PRINT " EXPERIMENT"
1030  NORMAL
9000  END
```

Lesson Two Self-Check Test #1

Directions — Match the words at the bottom of the page with the statements below and write the letter of your choice in the space provided. There are more answers than statements. The words can be used only once.

——— 1. To see an entire program you type ( ).

——— 2. The DOS command ( ) removes protection from a file on diskette.

——— 3. The command ( ) will cause the files on diskette to be displayed.

——— 4. The command ( ) will make a copy of a program on diskette and place it in the memory of the computer.

——— 5. The ( ) command is used to change the name of a file currently on diskette.

——— 6. If you wish to secure a file to protect it from being erased, renamed, or changed, you use the ( ) command.

——— 7. ( ) is the command that allows you to make notes to yourself within a BASIC program.

——— 8. The ( ) command is used to make a copy of a program currently in temporary memory and place it permanently on diskette in a file.

——— 9. The ( ) command tells the computer to stop looking for line numbers and stop processing the program.

——— 10. If you wish a message to be seen flashing on the screen you use the ( ) command.

——— 11. The ( ) command returns the computer to regular print mode of white letters on black.

——— 12. The ( ) command causes all subsequent print lines to print black letters on a white background.

——— 13. The ( ) command is used to slow down the rate at which the computer prints on the screen.

——— 14. The ( ) command allows you to remove a single file from diskette permanently.

A . CATALOG
B . COMMAND
C . DELETE
D . END
E . FLASH
F . INVERSE
G . LOAD
H . LOCK
I . LIST
J . NORMAL
K . REM
L . RENAME
M . SAVE
N . SPEED
O . SYNTAX
P . TYPE
Q . UNLOCK

Lesson Two Self-Check Test #2

Directions — If the statement is TRUE, write a "T" in the space provided. If the statement is FALSE, write an "F" in the space, and change the italicized portion to make the statement TRUE.

——— 1. It is good programming practice to begin a program with a ***REMARK***.

——— 2. This course uses ***BASIC*** commands to write programs.

——— 3. ***DOS*** (Disk Operating System) commands are used to control files, programs, or data on diskette.

——— 4. A good place to begin the main body of the program is on line ***1000***.

——— 5. When using INVERSE and FLASH commands you must use the command ***SAVE*** to return to regular print mode.

——— 6. The ***SPEED*** command can be used in a program as a special effects tool and to control the reading rate.

——— 7. The ***NORMAL*** command on a line by itself will print a blank line.

——— 8. ***String literals*** within quotation marks will be printed on the CRT exactly as written.

——— 9. In APPLESOFT the END command is ***always necessary*** within your program.

——— 10. To SAVE a program we file it under a ***number line***.

——— 11. During the process of saving a program, the flashing ***cursor*** disappears.

——— 12. If a file is locked on diskette an ***asterisk*** will appear in front of the catalog listing.

——— 13. If an attempt is made to rename a locked file on diskette the computer will ***erase the file*** and give you a message.

Lesson Two Self-Check Test #3

Directions — Read each statement and choose the most acceptable answer. Write the letter of your choice in the space provided.

——— 1. The PRINT command in conjunction with INVERSE, NORMAL, FLASH, and SPEED commands creates

A. a situation difficult to change.
B. a SYNTAX ERROR from the computer.
C. a more interesting print line.
D. a description of the program.

——— 2. The program line recommended for the SAVE command is

A. 10 C. 30 E. 50
B. 20 D . 40

——— 3. There are three rules when naming files on the APPLE II computer. The rule that is not necessary is

A. the file name must indicate the bytes used.
B. any character except a comma may appear in the name.
C. the file name must begin with a letter.
D. the file name must be from 1 to 30 characters in length.

——— 4. It is recommended that the main body of your program begins on line

A. 10 C. 110 E. 1010
B. 100 D. 1000

——— 5. The letter "A" that appears before a file name when you see the program stored on diskette means

A. after the actual writing.
B. APPLESOFT BASIC program.
C. added to the CATALOG.
D. ALL SAVED to use.

——— 6. The SYNTAX for the RENAME command is

A. RENAME current file name; new file name
B. RENAME; current file name new file name
C. RENAME current file name, new file name
D. RENAME; current file name, new file name

——— 7. The SYNTAX for deleting a file from diskette is

A. DELETE; file name
B. DELETE, file name
C. file name DELETE
D. DELETE file name

——— 8. This command will put an asterisk in front of the CATALOG listing of a file.

A. LIST C. SAVE E. PRINT
B. LOAD D . LOCK

——— 9. The line number recommended for a short description of your program is

A. 10 C. 1000
B. 20 D. 5000

——— 10. To insert a statement line into a program, type line number followed by instructions and

A. press ESC key.
B. press REPT key.
C. press CTRL key.
D. press SHIFT key.
E. press RETURN key.

——— 11. To see a portion of a program you type

A. LOAD 0010,0100
B. LIST 0010,0100
C. RUN 0010,0100
D. SAVE 0010,0100

——— 12. To delete a line from your program you

A. type line number and PRESS DELETE.
B. type line number and PRESS CTRL C.
C. type line number and PRESS RETURN.
D. PRESS RETURN and wait for the BELL.

——— 13. The slowest speed at which the APPLE computer prints is

A. 0 C. 255
B. 1 D . 100

——— 14. We number program lines within a section in increments of

A. 100 C . 10 E. 20
B. 50 D . 1000

——— 15. The SYNTAX for the SPEED command is

A. SPEED;n C. SPEED,N
B. SPEED = n D. N = SPEED
E. N,SPEED

——— 16. Any combination of letters, numbers, symbols, or spaces placed within quotation marks is called a

A. Remark Section.
B. PRINT Command.
C. Syntax Statement.
D. Structure Listing.
E. String Literal.

Assignment For Lesson Two

Your assignment for Lesson Two is to write a program which prints a narrative about your hobby or pet.

The program must include the following:

1. At least one line printed in INVERSE.
2. At least one line printed in FLASHing mode.
3. At least one line printed in NORMAL.
4. The SPEED command must be used at least two times: once to slow down the printing speed, and once to return the speed to normal, 255.
5. Line 20 should be a REM which includes the SAVE command.
6. Use the SAVE command to store your program in a file on diskette.

Additional Assignment For Lesson Two, #1

Write a program which alternates printing your name in INVERSE mode and then in FLASH mode until your name has been printed nine times on the screen, four times flashing, and five times in INVERSE.

Use the END command as the last line of the program.

Make lines 10 and 20 remarks to include a program description and the SAVE command for the program.

Additional Assignment For Lesson Two, #2

Write a program which demonstrates the difference between the print speeds of 0, 50, 100, 150, 200 and 255.

Use FLASH, INVERSE and NORMAL print modes in the program.

LESSON THREE: *VARIABLES*

LESSON OBJECTIVES

In this lesson you will be introduced to the two types of variables used in BASIC programming. *Numeric variables* store numbers. *String variables* store characters. The numbers or characters stored in variables are called the variable's *value*.

You will learn two ways to assign values to variables: 1) the programmer may assign a value directly to a variable, or 2) the operator may enter the value of a variable from the keyboard.

The new BASIC commands introduced in this lesson are: **LET** and **INPUT.**

Program Listing For Lesson 3

Assigning Variables

```
1000 A$ = "HORSE"
1010 A$ = "RIDER"
1020  PRINT A$
1030  END
```

Prompted Input: Version #1

```
1000  PRINT "ENTER NAME."
1010  INPUT N$
1020  HOME
1030  PRINT "HELLO!!"
1040  PRINT N$
1050  END
```

Prompted Input: Version #2

```
1000  INPUT "ENTER NAME ";N$
1020  HOME
1030  PRINT "HELLO!!"
1040  PRINT N$
1050  END
```

Prompted Input: Final Version

```
10  REM :: INPUT EXAMPLE PGM
20  REM  SAVE PROMPTED INPUT
1000  INPUT "ENTER NAME. ";N$
1020  HOME
1030  PRINT "HELLO!!"
1040  PRINT N$
1050  INPUT "ENTER AGE ";AGE
1060  PRINT "IT MUST BE FUN TO BE"
1070  PRINT AGE
1080  END
```

Lesson Three Self-Check Test #1

Directions — For statements one through thirteen you are to use the following key. For each statement select the word that goes with the statement. Write the letter of the word in the space provided. The items can be used more than once.

A. PRINT	C. LET
B. INPUT	D. VARIABLE(S)

—— 1. In programming this stands for characters, words, whole sentences, or numbers.

—— 2. Allows the operator to enter information which can be used within the program.

—— 3. This is usually implied rather than actually typed.

—— 4. When used on a line by itself this command instructs the computer to display a blank line on the CRT.

—— 5. Displays a string literal or the value of a variable on the CRT.

—— 6. A Prompted Input, consisting of the INPUT command, a string literal, and a variable, will replace this command and save one programming line.

—— 7. There are two types; one for numerics, and one for strings.

—— 8. This command allows the programmer to assign values to variables.

—— 9. Information is entered from the keyboard and stored with this command.

—— 10. These cannot contain blanks, reserved words, or punctuation marks.

—— 11. Its value can be changed.

—— 12. Will retain only the last information stored in it.

—— 13. To the computer this is a place in the memory of the machine in which we store information.

Lesson Three Self-Check Test #2

Directions — You are to match the answers at the bottom of the page with the numbered statements. Write the letter of your choice in the space provided. There are more statements than answers. An answer may be used more than once.

—— 1. A ( ) must have a name that begins with a letter of the alphabet.

—— 2. If it is a character variable the name must end with a ( ).

—— 3. A letter followed by a ( ) or a character is an acceptable numeric variable.

—— 4. The computer recognizes the first ( ) characters in a variable name.

—— 5. The two BASIC commands used to assign variables are the LET command and the ( ) command.

—— 6. A character string is enclosed in ( ).

—— 7. To display the value of a variable named A$ we use the command ( ) A$.

—— 8. One who writes and debugs programs is called the ( ).

—— 9. The computer displays a question mark when waiting for a response from the operator. The question mark is called a ( ).

—— 10. When assigning a value to a numeric variable you do not enclose the value in ( ).

—— 11. If you attempt to assign a character to a numeric variable (B = "3") the computer will give you a message of TYPE ( ) ERROR.

—— 12. BX would be an acceptable name for a ( ) variable.

—— 13. One who uses programs is called the ( ).

—— 14. The character or number stored within a variable is called the ( ) of the variable.

—— 15. A Prompted Input saves ( ) lines(s) of writing for programmers.

—— 16. The ( ) command allows programmers to assign values to variables within the BASIC programming line.

—— 17. B3$ would be an acceptable name for a ( ) variable.

—— 18. NUMB would be an acceptable name for a ( ) variable.

A. CHARACTER
B. DOLLAR SIGN
C. INPUT
D. MISMATCH
E. NUMBER
F. NUMERIC
G. ONE
H. OPERATOR
I. PRINT
J. PROGRAMMER
K. PROMPT
L. QUOTATION MARKS
M TWO
N. VALUE
O. VARIABLE
P. SPEED
Q. LET

Assignment For Lesson Three

Write a data entry program which includes the following information:

In line 500, assign the string literal "THANK YOU" to a character variable.

Using prompted INPUTs, have the program instruct the operator to enter a name, an address and a zip code.

Store the name and address in character variables, and store the zip code in a numeric variable.

In response to the name entry, have the computer PRINT "THANK YOU", and then PRINT the name.

In response to the address entry, have the computer PRINT "THANK YOU", "YOU LIVE AT", and then PRINT the address.

In response to the zip code entry, have the computer PRINT "THANK YOU", "YOUR ZIP CODE IS", and then PRINT the zip code.

Include a program title in a REMark at the beginning of the program and the SAVE command in a REMark in line 20.

Additional Assignment For Lesson Three, #1

This assignment is to write a small data entry program.

After the opening remark lines assign the string literals "YOUR NEIGHBOR'S NAME IS" and "YOUR DOCTOR'S NAME IS" to two variables. (You may use other people you know if you wish.)

Using two INPUT commands, have the operator enter his or her doctor's and neighbor's names.

Clear the screen and print the variable containing the first string literal "YOUR NEIGHBOR'S NAME IS"

On the next line PRINT the neighbor's name in FLASHing print mode.

PRINT the variable containing the second string literal.

PRINT the doctor's name in FLASHing mode.

Then return the computer to NORMAL print mode and end the program.

Additional Assignment For Lesson Three, #2

Write a program using the LET and INPUT commands which instructs the computer to respond to various operator inputs, simulating conversation.

LESSON FOUR: *MATH & FORMATTED PRINTING*

LESSON OBJECTIVES

Lesson Four includes two parts: first, you will be introduced to the *calculating* capabilities of the computer; second, you will be introduced to the *screen formatting* capabilities of the computer.

The symbols used in computer calculations are: *"+" (addition), "-" (subtraction), "*" (multiplication), "/" (division), "^" (exponentiation or repeated multiplication), "=" (equals),* and *"( )" (parentheses).*

If more than one calculation symbol is used in a single line, the computer performs the calculations in a particular order, called a *hierarchy*.

Parentheses may be used to alter the order in which calculations are performed.

An explanation of "Scientific Notation", *(E-Notation)*, is included in the back of the Study Guide.

The second part of the lesson introduces you to the screen formatting capabilities of the computer. To format the CRT is to arrange the printing on the 40 column, 24 line *text screen.*

The computer uses two punctuation marks and three commands to format the CRT.

The two formatting punctuation marks are the *comma (,)* and the *semicolon (;).*

The new BASIC commands introduced in this lesson are: **TAB( ), HTAB,** and **VTAB.**

Program Listing For Lesson 4

Addition Example

```
1000 A = 5 + 2
1010  PRINT A
```

Subtraction Example

```
1000 S = 322 - 184
1010  PRINT S
```

Multiplication Example

```
1000 M = 13 * 12
1010  PRINT M
```

Division Example

```
1000 D = 1024 / 128
1010  PRINT D
```

Exponentiation Example

```
1000 X = 5 * 5 * 5 * 5 * 5 * 5
1010  PRINT X
1020 E = 5 ^ 6
1030  PRINT E
```

Computer Math

```
10  REM :: 3.3 CALCULATION
20  REM  SAVE COMPUTER MATH
500 C = 3.3
1000  HOME
1010  INVERSE
1020  PRINT " COMPUTER MATH"
1030  NORMAL
1040  PRINT
1050  PRINT
1060  INPUT "ENTER NUMBER ";N
1070  PRINT
1080 A = N + C
1090  PRINT "YOUR NUMBER ADDED TO 3.3 IS"
1100  PRINT A
1110  PRINT
1120 D = N / C
1130  PRINT "YOUR NUMBER DIVIDED BY 3.3 IS"
1140  PRINT D
1150  END
```

Comma Program, #1

```
1000  PRINT 1,17,33
1010  PRINT "ZONE 1","ZONE 2","ZONE 3"
```

Comma Program, #2

```
1000  PRINT ,"ZONE 2","ZONE 3"
1010  PRINT ,,"ZONE 3"
```

Semicolon

```
10  REM :: SEMICOLON EXAMPLE PGM
20  REM  SAVE SEMICOLON
500 A$ = "APPLES"
1000  PRINT "WE HAVE ";A$;" FOR SALE"
1010  INPUT "HOW MANY DO YOU WANT? ";N
1020 A = N * .25
1030  PRINT N;" ";A$;" WILL COST $";A
1040  END
```

VTAB Demonstration

```
1000  HOME
1010  VTAB 10
1020  PRINT "ON LINE 10"
```

TAB-HTAB: Version #1

```
10  REM  TAB-HTAB EXAMPLE
20  REM  SAVE TAB-HTAB
500  SPEED= 0
1000  HOME
1010  VTAB 2
1020  HTAB 20
1030  PRINT "* 1ST HTAB";
1040  HTAB 1
1050  PRINT "* 2ND HTAB"
```

TAB-HTAB: Final Version

```
10  REM  TAB-HTAB EXAMPLE
20  REM  SAVE TAB-HTAB
500  SPEED= 0
510  INVERSE
1000  HOME
1010  VTAB 2
1020  HTAB 20
1030  PRINT "* 1ST HTAB";
1040  HTAB 1
1050  PRINT "* 2ND HTAB"
1060  VTAB 4
1070  PRINT  TAB( 5);"* 1ST TAB"; TAB( 25);"* 2ND TAB"
1080  SPEED= 255
1090  NORMAL
1100  END
```

Lesson Four Self-Check Test #1

Directions — This test is divided into 3 parts.

Part One Directions — In the space provided, write the symbol for the five basic math functions used by the computer.

——— ADDITION ——— MULTIPLICATION ——— DIVISION

——— SUBTRACTION ———EXPONENTIATION

Part Two Directions — The computer performs mathematical calculations with the rules of hierarchy used to solve the formulas in higher mathematics. You are to put the calculations in the order they are performed by the computer. Place the number of the order in the space provided.

——— Addition and Subtraction
——— Calculations enclosed in parentheses
——— Exponentiation
——— Multiplication and Division

Part Three Directions — Listed below are some calculation problems. Write them in the mathematical form you would use in one of your programs. Show the steps in the order the computer would process the calculations.

1. N equals four plus eight to the second power, divided by eight minus one:

2. N equals, (twenty-four divided by six plus three), divided by (three plus four):

3. N equals (five multiplied by two plus (six minus two)) divided by seven, add three to the third power, add two.

Lesson Four Self-Check Test #2

Directions — If the statement is TRUE, place a "T" in the space provided. If the answer is FALSE, place an "F" in the space and replace the italicized portion to make the statement TRUE.

——— 1. ***Exponentiation*** means "raise to the power of".

——— 2. With exponentiation you ***divide*** a number by itself a given number of times.

——— 3. Results of a math calculation can be stored in a ***variable*** and used throughout a program.

——— 4. Value can be assigned to a variable by using the ***math calculations.***

——— 5. The computer processes calculations from ***right to left*** and will perform certain math operations before others.

——— 6. A programmer can use ***parentheses*** to change the order in which the computer performs calculations.

——— 7. To format the CRT means to arrange the ***printing*** on the screen.

——— 8. On the APPLE II the text screen has ***25*** lines and is capable of printing 40 characters on a single line.

——— 9. If you attempt to print a message longer than 40 characters the computer will ***halt and give you an error message.***

——— 10. Each 40 character screen line is divided into ***three*** print zones.

——— 11. Sometimes a computer will type out ***numbers*** in scientific notation or E-Notation.

——— 12. If you print beyond ***255*** characters the computer will print a back slash indicating it has cancelled the entire ***255*** characters.

——— 13. The print line used for printed reports is usually 80 or ***136*** characters long.

——— 14. Home position on the screen is ***line 2, position 2.***

Lesson Four Self-Check Test #3

Directions — For each statement below select the word(s) from the key that completes the statement. Write the letter of your choice(s) in the space provided. Items can be used more than once. Some will not be used at all.

A . COMMA(S)
B . HTAB
C . QUESTION MARKS
D . PERIOD(S)
E . PRINT
F . SEMICOLON(S)
G . TAB
H . VTAB

——— 1. The ( ) in a PRINT statement will instruct the computer to print in the next print zone.

——— 2. The ( ) when used within a PRINT statement to format the CRT, allows you to print character strings, variables, or numbers all together on the same line.

——— 3. The ( ) allows you to select on which of the 24 lines the computer will print.

——— 4. This punctuation mark, the ( ), can be used to skip one or more print zones.

——— 5. The ( ) and ( ) commands allow you to determine at which position on the line the computer will PRINT.

——— 6. The ( ) is the least used of the formatting techniques.

——— 7. The ( ) command is used within a PRINT statement.

——— 8. In a PRINT statement, the ( ) at the end of a string literal instructs the computer to stay on the same line for next printing.

——— 9. ( ) and ( ) are used on a separate line to establish the position of the next PRINT statement.

——— 10. ( ) instructs the computer to print blanks from the current cursor position to the position specified within parentheses.

——— 11. ( ) positions the cursor directly on a print line.

——— 12. The ( ) is the most often used of the punctuation marks when formatting the screen.

——— 13. You would ( ) first to pick the line on which to PRINT.

——— 14. You cannot ( ) backwards.

——— 15. ( ) positions the cursor directly and does not erase a portion of the display.

——— 16. The ( ) command's position must be enclosed in parentheses.

——— 17. You can mix ( ) with string literals within a PRINT statement by using semicolons; more than one can be used on a line.

Math Assignment For Lesson Four

Write a program which utilizes the math calculations.

1. Use a prompted INPUT to allow the operator to enter a number.
2. Multiply the number by 5.64.
3. Subtract six from the number.
4. Raise the number to the fourth power.
5. In a multiple calculation, divide the number by seven and add 204 to the number.
6. After each calculation, use a PRINT statement to describe the calculation and the results of the calculation.

For example, following the first multiplication calculation, the program should PRINT "YOUR NUMBER TIMES 5.64 IS". Then the program should PRINT the product of the calculation.

Formatted Printing Assignment For Lesson Four

Your second assignment is to write a program using the CRT formatting techniques.

1. Clear the screen and position the cursor at row 1, position 1.
2. PRINT the program title "NAME ENTRY PROGRAM" at screen line three, position 10.
3. On screen lines five, six and seven, PRINT the words: "NAME", "AGE" and "PHONE".
4. On line ten instruct the operator to enter his or her name and PRINT the name at line five, position ten.
5. On line eleven instruct the operator to enter his or her age and PRINT the age at line six, position ten.
6. On line twelve instruct the operator to enter his or her telephone number and PRINT it at line seven, position ten.

After the operator has entered name, age and telephone number:

7. PRINT "THANK YOU" and the operator's name at line fifteen, position five.
8. On line nineteen, PRINT "IN 10 YEARS YOU WILL BE" and the operator's age in ten years.
9. On line twenty-two PRINT "PROGRAM" in print zone one and "ENDED" in print zone two.
10. You must use each of the formatting techniques at least once: the comma, the semicolon, TAB( ) HTAB and VTAB.

Additional Assignment For Lesson Four, #1

The following program has three errors. Find and correct each of the errors.

```
10  REM  :: LESSON 4 ADDITIONAL 1
20  REM   SAVE LES4A1
1000  HOME
1010  VTAB 20
1020  INPUT "ENTER YOUR AGE ";A
1030  VTAB 2
1040  PRINT "YOU ARE ";A;" YEARS OLD"
1050  VTAB 25
1060  INPUT "ENTER YOUR NAME ";N
1070  VTAB 3
1080  PRINT "YOUR NAME IS ";N$
1090  VTAB 20
1100  INPUT "ENTER YOUR FAVORITE COLOR ";C$
1110 XTAB4
1120  PRINT "YOUR FAVORITE COLOR IS ";C$
1130  END
```

Additional Assignment For Lesson Four, #2

In this assignment you are to write a program which uses the PRINT formatting commands: VTAB, HTAB, TAB( ), the comma "," and the semicolon ";".

The program should demonstrate the function of each formatting technique.

LESSON FIVE: *DECISIONS*

LESSON OBJECTIVES

In Lesson Five you are introduced to the decision making capabilities of the computer through the various uses of the **IF-THEN** command pair.

You will learn how to alter the *flow* of the program using the **GOTO** command and the **IF-THEN** command. (Program flow refers to the order in which the computer processes commands.)

You will learn how to create a program *loop* and how to interrupt the execution of a program using **CTRL-C** and **CTRL-RESET.**

You will learn how to use the **IF-THEN** command to condition the execution of any command.

You will learn how to set up and test for conditions which compare the values of variables using the equivalency symbols, *"="(equal to), "<" (less than), ">" (greater than)* and *"<>" (does not equal).*

You will learn how to test for more than one condition using **AND** and **OR.**

The new BASIC commands introduced in this lesson are: **GOTO, IF-THEN, AND** and **OR.**

Program Listing For Lesson 5

GOTO Program: Version #1

```
10  REM  :: GOTO PROGRAM
20  REM  SAVE GOTO TEST
1000  HOME
1010  PRINT "HELLO. HERE I AM AGAIN"
1020  PRINT "MY NAME IS BEN"
9000  PRINT "......END....."
9010  END
```

GOTO Program: Version #2

```
10  REM  :: GOTO PROGRAM
20  REM  SAVE GOTO TEST
50  GOTO 9000
1000  HOME
1010  PRINT "HELLO. HERE I AM AGAIN"
1020  PRINT "MY NAME IS BEN"
9000  PRINT "......END....."
9010  END
```

GOTO Program: Final Version

```
10  REM :: GOTO PROGRAM
20  REM  SAVE GOTO TEST
1000  HOME
1010  PRINT "HELLO. HERE I AM AGAIN"
1020  PRINT "MY NAME IS BEN"
1030  GOTO 1010
9000  PRINT "......END......"
9010  END
```

IF-THEN Example: Version #1

```
10  REM  :: CONTROLLING PROGRAM FLOW
20  REM  SAVE IF-THEN EXAMPLE
1000  HOME
1010  INPUT "ENTER A WORD ";W$
1100  PRINT "THE WORD YOU ENTERED WAS ";W$
1110  GOTO 1010
9000  END
```

IF-THEN Example: Final Version

```
10  REM :: CONTROLLING PROGRAM FLOW
20  REM  SAVE IF-THEN EXAMPLE
1000  HOME
1010  INPUT "ENTER A WORD OR 'STOP' TO END ";W$
1020  IF W$ = "STOP" THEN  GOTO 9000
1100  PRINT "THE WORD YOU ENTERED WAS ";W$
1110  GOTO 1010
9000  END
```

IF-THEN Print

```
10  REM :: ANOTHER IF-THEN
20  REM  SAVE IF-THEN-PRINT
1000  HOME
1010  INPUT "DO YOU LIKE TO PROGRAM (YES OR NO) ";AN$
1020  IF AN$ = "YES" THEN  PRINT "THAT'S WONDERFUL"
1030  IF AN$ = "NO" THEN  PRINT "I'M SORRY TO HEAR IT"
2000  END
```

Number IF-THEN

```
10  REM :: IF-THEN WITH =,<,> ::
20  REM  SAVE NUMBER IF-THEN
1000  HOME
1010  INPUT "ENTER YOUR AGE ... OR Ø TO END ";AGE
1020  IF AGE = Ø THEN 9000
1030  IF AGE < 21 THEN  PRINT "THE ENERGY OF YOUTH"
1040  IF AGE > 20 THEN  PRINT "THE WISDOM OF AGE"
1050  GOTO 1010
9000  PRINT "GOODBYE"
9010  END
```

Not Equal

```
10  REM  THE DOES NOT EQUAL SIGN
20  REM  SAVE NOT EQUAL
1000  HOME
1010  PRINT "IS EVERYBODY HAPPY?"
1020  INPUT "(ANSWER YES IF HAPPY) ";A$
1030  REM :: TEST ANSWER ::
1040  IF A$ = "YES" THEN  PRINT "THAT'S GREAT"
1050  IF A$ <  > "YES" THEN  PRINT "TOO BAD"
9000  END
```

AND/OR Demonstration

```
1000  INPUT "ENTER A NUMBER BETWEEN 1 AND 4: ";N
1010  IF N < 1 OR N > 4 THEN  PRINT "WRONG NUMBER"
1020  IF N > 0 AND N < 5 THEN  PRINT "RIGHT NUMBER"
```

Multiple Choice Example

```
10  REM :: MULTIPLE CHOICE EXAMPLE
20  REM  SAVE MULTIPLE-CHOICE
500  HOME
1000  PRINT "WHY DO FOOTBALL PLAYERS WEAR PADS?"
1010  PRINT
1020  PRINT "1. THEIR SHIRTS ARE TOO BIG.."
1030  PRINT "2. THEY WANT TO LOOK LARGER.."
1040  PRINT "3. PADS KEEP THEM WARM IN THE WINTER .."
1050  PRINT "4. THE PADS PROTECT THEM.."
1060  PRINT
1070  INPUT "ENTER 1-4 ";N
1080  REM  :: CHECK ANSWERS ::
1090  IF N = 1 THEN  PRINT "NO! THEIR SHIRTS FIT FINE.."
1100  IF N = 2 THEN  PRINT "NO! THEY ARE LARGE ENOUGH.."
1110  IF N = 3 THEN  PRINT "NO! THEY STAY WARM WHEN THEY RUN.."
1120  IF N = 4 THEN  PRINT "YES! THAT IS CORRECT.."
1130  IF N <  > 4 THEN  GOTO 1060
1140  PRINT "---- GOODBYE ----"
1150  END
```

Lesson Five Self-Check Test #1

Directions — Fill in the blanks. Provide the missing word(s) by writing it in the space provided.

1.———————— The ( ) command instructs the computer to jump lines within a program to a specific line where it continues to process the program.

2.———————— ( ) tells the computer to interrupt the program execution and to tell you on what line it was interrupted.

3.———————— By using an ( ) command you can exit from a loop and end the program without using the special control keys.

4.———————— If you place a ( ) on the line to which the computer returns repeatedly you can allow the operator to enter information into the computer a number of times. This type of loop is used frequently in data entry programs.

5.———————— The ( ) command can be used to conditionally execute any of the BASIC commands.

6.———————— The ( ) command can be used toward the end of the program to instruct the computer to return to a smaller line number.

7.———————— The ( ) command allows you to make tests in which two or more conditions must be true before the instructions following the THEN will be processed.

8.———————— The ( ) command allows you to make tests in which any of two or more conditions must be true before the instruction following the THEN will be executed.

9.———————— ( ) will stop the program in progress but will not give the line number it was executing when it was interrupted.

10.———————— If the program processes a section of the program over and over with no way to get out except with a control key, it is called a(n) ( ).

Lesson Five Self-Check Test #2

Directions — To make mathematical comparisons, the BASIC language uses standard math symbols. In the blanks write the keyboard symbol which stands for the italicized words.

——— If A ***is equal to*** B then (instructions)
——— If A ***is less than*** B then (instructions)
——— If A ***is greater than*** B then (instructions)
——— If A ***is not equal to*** B then (instructions)

Lesson Five Self-Check Test #3

Directions — A series of commands are listed below. "A" has been given the value of 5 and "B" the value of 6: A=5, B=6. Make the comparisons below and circle the instructions following the THEN only if the conditions are met.

1. IF A<>B THEN PRINT "YES"
2. IF A > B THEN PRINT "YES"
3. IF A<B THEN PRINT "YES"
4. IF A>0 AND B<5 THEN PRINT "YES"
5. IF A<6 OR B>7 THEN PRINT "YES"
6. IF A>0 AND B>7 THEN PRINT "YES"
7. IF A<5 OR B>6 THEN PRINT "YES"
8. IF A=B THEN PRINT "YES"
9. IF A<5 OR B>2 THEN PRINT "YES"
10. IF A=5 AND B>1 THEN PRINT "YES"

Assignment For Lesson Five

Your assignment for this lesson is to write a multiple choice quiz on any subject. Include one or more questions in your quiz.

For each question give the operator four possible answers from which to choose: one correct answer, and three incorrect answers.

Use IF-THEN commands to test the operator's answer to each question.

Instruct the computer to PRINT an appropriate response to each correct answer.

Additional Assignment For Lesson Five, #1

Frequently the IF-THEN command is used to allow an operator to end a program on demand.

This assignment is to write a program that allows the operator to enter a name.

If the name entered is "STOP" you are to end the program.

If the name entered is not the word "STOP", you are to print the name on screen line 12 beginning in column 20, again on line 16 beginning in position 5, and again on line 5 in print zones one and two. On line 20 print the name three times using the semicolon.

When the name has been printed in its numerous positions on the screen, you should again allow the operator to enter a word.

Make the program continue in this loop until the operator enters the word "STOP".

Additional Assignment For Lesson Five, #2

Using the INPUT and IF-THEN commands you can make the computer ask the operator questions and then respond to the answers.

Write a program to ask the operator at least three questions which can be answered "YES" or "NO".

After each question use the IF-THEN command to PRINT an appropriate response.

LESSON SIX: *PROGRAMMING TECHNIQUES*

LESSON OBJECTIVES

In Lesson Six you will be introduced to several short cuts and techniques which may be used in Applesoft BASIC when writing programs, and you will be introduced to the concept of *"Modular Programming"* which may be used when writing long programs.

You will be introduced to the following programming techniques: 1. the *question mark (?)* as shorthand for the PRINT command; 2. the *"Dummy Input";* 3. *Multiple Statement lines*; 4. the **RUN** command followed by a line number to debug programs; 5. the use of **"CTRL-S"** to scan a long program listing; 6. the use of **CHR$(7)** to make the computer's bell sound; 7. the semicolon may be omitted when printing variables and string literals on the same line; 8. the use of the **DEL** command to delete a group of statement lines; and 9. the use of a *multiple*-**INPUT** command.

The new BASIC commands introduced in this lesson are: **DEL** and **CHR$(7).**

After demonstrating the program writing techniques, you will be introduced to the concepts of *"Modular Programming"* to write a long program. Modular programming is the technique of writing long programs as a series of short program sections (or modules).

To design a long program, you first *outline the program* and then write a short *logic paragraph* describing each module or operation.

The "NUMBER-GUESS" program flow outline listed below will be covered during the lecture on modular programming. It is not necessary to read the outline prior to viewing.

Number Guess Program Flow Outline

"NUMBER-GUESS" is a game for two players in which one player enters a number and the other tries to guess it.

Section #1 — HOUSEKEEPING, identifies the program and is used for any other set-up needed before the main section.

Section #2 — TITLE PAGE, displays the title page and introduces the players.

Section #3 — INSTRUCTIONS, explains the rules of the game.

Section #4 — PLAY OF GAME, is the actual play of the game.

Section #5 — END OF GAME, is the ending section of the game.

Number-Guess Logic Paragraphs

Section #1 — HOUSEKEEPING.
Identify program in REMarks and assign bell to "BL$".

Section #2 — TITLE PAGE.
Display game title in INVERSE and use prompted INPUTs to assign player names to "P1$" and "P2$".

Section #3 — INSTRUCTIONS.
Use PRINT commands to explain the play of the game and use dummy input to hold instructions on screen.

Section #4 — PLAY OF GAME.
First use prompted INPUT to allow player one to enter number between 1 and 100. Store number in variable "N". Use IF-THEN with "OR" to verify for valid number. Sound bell when entered.

Second, use prompted INPUT to allow player two to guess the number and store guess in variable "G".

Use IF-THEN's to test player two's guess. If guess is zero, player two gives up. So, PRINT "PLAYER ONE WINS", FLASHing, use two bells, and end program. If guess is too low, PRINT "TOO LOW" and let player two guess again. If guess is too high, PRINT "TOO HIGH" and let player two guess again.

By process of elimination, if guess is not zero, and guess is not too low, and guess is not too high, then guess is correct. So, congratulate player one, FLASH, using three bells and end program.

Section #5 — END OF GAME.
Ask players if they want to play again: "Y" for yes, "N" for no. If yes, restart game at title page. If other than no, ask again. If no, PRINT "IT WAS FUN", in INVERSE, use one bell, and end program.

Program Listing For Lesson 6

Special Program: Version #1

```
10  REM :: SPECIAL PROGRAM
20  REM   SAVE SPECIAL PROGRAM
1000  HOME
1010  VTAB 5
1020  HTAB 5
1030  INVERSE
1040  PRINT "A SPECIAL PROGRAM"
1050  NORMAL
1060  VTAB 8
1070  PRINT "TO HOLD A MESSAGE ON THE SCREEN"
1080  PRINT "UNTIL THE OPERATOR HAS READ IT, "
1090  PRINT "WE CAN USE A DUMMY INPUT"
1100  PRINT "WHICH WILL WAIT UNTIL"
1110  PRINT "THE OPERATOR PRESSES RETURN"
1120  VTAB 20
1130  INPUT "(PRESS RETURN TO CONTINUE) ";A$
```

Special Program: Version #2

```
10  REM :: SPECIAL PROGRAM
20  REM   SAVE SPECIAL PROGRAM
1000  HOME : VTAB 5: HTAB 5: INVERSE
1040  PRINT "A SPECIAL PROGRAM"
1050  NORMAL : VTAB 8
1070  PRINT "TO HOLD A MESSAGE ON THE SCREEN"
1080  PRINT "UNTIL THE OPERATOR HAS READ IT, "
1090  PRINT "WE CAN USE A DUMMY INPUT"
1100  PRINT "WHICH WILL WAIT UNTIL"
1110  PRINT "THE OPERATOR PRESSES RETURN"
1120  VTAB 20
1130  INPUT "(PRESS RETURN TO CONTINUE) ";A$
```

Special Program: Version #3

```
10  REM :: SPECIAL PROGRAM
20  REM   SAVE SPECIAL PROGRAM
1000  HOME : VTAB 5: HTAB 5: INVERSE
1040  PRINT "A SPECIAL PROGRAM"
1050  NORMAL : VTAB 8
1070  PRINT "TO HOLD A MESSAGE ON THE SCREEN"
1080  PRINT "UNTIL THE OPERATOR HAS READ IT, "
1090  PRINT "WE CAN USE A DUMMY INPUT"
1100  PRINT "WHICH WILL WAIT UNTIL"
1110  PRINT "THE OPERATOR PRESSES RETURN"
1120  VTAB 20
1130  INPUT "(PRESS RETURN TO CONTINUE) ";A$
2000  HOME
2010  VTAB 10: HTAB 5
2020  INPUT "ENTER A NUMBER (0 TO END): ";N
2030  IF N > 0 THEN  HOME : PRINT "YOUR NUMBER IS ";N: GOTO 2010
2040  END
```

Special Program: Version #4

```
10  REM :: SPECIAL PROGRAM
20  REM   SAVE SPECIAL PROGRAM
1000  HOME : VTAB 5: HTAB 5: INVERSE
1040  PRINT "A SPECIAL PROGRAM"
1050  NORMAL : VTAB 8
1070  PRINT "TO HOLD A MESSAGE ON THE SCREEN"
1080  PRINT "UNTIL THE OPERATOR HAS READ IT, "
1090  PRINT "WE CAN USE A DUMMY INPUT"
1100  PRINT "WHICH WILL WAIT UNTIL"
1110  PRINT "THE OPERATOR PRESSES RETURN"
1120  VTAB 20
1130  INPUT "(PRESS RETURN TO CONTINUE) ";A$
2000  HOME
2010  VTAB 10: HTAB 5
2020  INPUT "ENTER A NUMBER (0 TO END): ";N
2030  IF N > 0 THEN  HOME : PRINT "YOUR NUMBER IS ";N: GOTO 2010
2100 A = 10:G$ = "GLASS"
2110  PRINT A;" BUILDINGS OF ";G$
2120  PRINT A" BUILDINGS OF "G$

3000  VTAB 15
3010  PRINT "ENTER TWO NUMBERS SEPARATED BY COMMAS"
3020  INPUT "ENTER NUMBERS: ";X,Y
3030  PRINT "YOU ENTERED "X" AND "Y

3100  PRINT  CHR$ (7)
3110  PRINT "THE BELL SOUNDS LIKE"; CHR$ (7)
3120 BL$ =  CHR$ (7): PRINT BL$
3130  PRINT "AND CAN BE ASSIGNED TO A VARIABLE";BL$
```

Special Program: Final Version

```
10  REM :: SPECIAL PROGRAM
20  REM  SAVE SPECIAL PROGRAM
1000  HOME : VTAB 5: HTAB 5: INVERSE
1040  PRINT "A SPECIAL PROGRAM"
1050  NORMAL : VTAB 8
1120  VTAB 20
1130  INPUT "(PRESS RETURN TO CONTINUE) ";A$
2000  HOME
2010  VTAB 10: HTAB 5
2020  INPUT "ENTER A NUMBER (0 TO END): ";N
2030  IF N > 0 THEN  HOME : PRINT "YOUR NUMBER IS ";N: GOTO 2010
2100 A = 10:G$ = "GLASS"
2110  PRINT A;" BUILDINGS OF ";G$
2120  PRINT A" BUILDINGS OF "G$
3000  VTAB 15
3010  PRINT "ENTER TWO NUMBERS SEPARATED BY COMMAS"
3020  INPUT "ENTER NUMBERS: ";X,Y
3030  PRINT "YOU ENTERED "X" AND "Y
3100  PRINT  CHR$ (7)
3110  PRINT "THE BELL SOUNDS LIKE"; CHR$ (7)
3120 BL$ =  CHR$ (7): PRINT BL$
3130  PRINT "AND CAN BE ASSIGNED TO A VARIABLE";BL$
```

Number Guess

```
10  REM :: NUMBER GUESSING GAME
20  REM SAVE NUMBER-GUESS
100 BL$ =  CHR$ (7)
999  REM  :: SET UP THE GAME ::
1000  HOME : VTAB 5: INVERSE
1010  PRINT " **** NUMBER-GUESS **** "
1020  NORMAL : PRINT
1030  PRINT "... A GAME FOR TWO PLAYERS"
1040  VTAB 10
1050  INPUT "PLAYER #1, ENTER YOUR NAME: ";P1$
1060  INPUT "PLAYER #2, ENTER YOUR NAME: ";P2$
1090  REM :: EXPLAIN THE RULES
1100  HOME : INVERSE : HTAB 10: PRINT " INSTRUCTIONS ": NORMAL
1110  VTAB 4: PRINT "IN THIS GAME, ";P1$;" WILL ENTER"
1120  PRINT "A NUMBER BETWEEN 1 AND 100."
1130  PRINT : PRINT P2$;" WILL TRY TO GUESS THE NUMBER."
1140  PRINT : PRINT "THE COMPUTER WILL REFEREE."
1150  VTAB 15: PRINT P2$;", YOU MUST LOOK AWAY"
1160  PRINT "WHILE ";P1$;" ENTERS A NUMBER."
1170  VTAB 20: INPUT "PRESS RETURN TO BEGIN PLAY. ";A$
1990  REM :: P1$ SELECTS NUMBER
2000  HOME : VTAB 8
2010  PRINT P1$", ENTER A NUMBER FROM 1 TO 100."
2020  INPUT "PLEASE USE WHOLE NUMBERS ONLY: ";N
2030  IF N < 1 OR N > 100 THEN 2000
2040  PRINT BL$
2990  REM :: P2$ GUESSES NUMBER
3000  HOME
3010  VTAB 4: PRINT P2$;", GUESS MY NUMBER."
3020  INPUT "(OR ENTER 0 TO GIVE UP): ";G
```

```
3030  IF G = 0 THEN 4000
3040  IF G < N THEN  HOME : VTAB 12: PRINT G;" WAS TOO SMALL":
      GOTO 3010
3050  IF G > N THEN  HOME : VTAB 12: PRINT G;" WAS TOO LARGE":
      GOTO 3010
3190  REM :: PLAYER 2 WINS
3200  HOME : PRINT BL$: VTAB 8: FLASH
3210  PRINT " ***** CONGRATULATIONS *****";BL$
3220  NORMAL : VTAB 12
3230  PRINT P2$;" GUESSED THE NUMBER"
3240  PRINT "IT WAS ";N;BL$
3250  GOTO 9000
3990  REM :: PLAYER 2 GIVES UP
4000  HOME : PRINT BL$: VTAB 4: FLASH
4010  PRINT " **** ";P1$;" WINS **** "
4020  NORMAL : VTAB 8: PRINT BL$;" THE NUMBER WAS ";N
8990  REM :: END OR PLAY AGAIN?
9000  VTAB 18: PRINT "DO YOU WANT TO PLAY AGAIN?"
9010  INPUT "(ENTER Y OR N): ";AN$
9020  IF AN$ = "Y" THEN 1000
9030  IF AN$ <  > "N" THEN 9000
9040  PRINT : INVERSE
9050  PRINT " IT WAS FUN ";BL$: NORMAL
9060  END
```

Lesson Six Self-Check Test #1

Directions — Match the item listed at the bottom of the page with the statement. Place the letter of your choice in the space provided. The terms may be used more than once.

——— 1. Using a question mark can be a shorthand way to write this command.

——— 2. What is it sometimes called when an INPUT statement is used to stop the program until the operator is ready to proceed?

——— 3. When an INPUT statement is used to stop the program what must the operator do to continue?

——— 4. What is a programming line with more than one command called?

——— 5. You can use more than one command on a line if you separate them with which punctuation mark?

——— 6. Most forms of BASIC require the use of this punctuation mark when printing variables and literals next to each other on the same line.

——— 7. If you are using multiple inputs on one line, what punctuation mark must separate the variables?

——— 8. To instruct the computer to make its "bell" sound, this command must be used with a PRINT command.

——— 9. These two keys are used to scan a long program.

——— 10. If you wish to see a portion of your program to change it, you use this command.

A . CHR$(7)
B . CTRL-C
C . CTRL-S
D . COLON(S)
E . COMMA(S)
F . DUMMY INPUT
G . LIST LINE NUMBER, LINE NUMBER
H . MULTIPLE STATEMENT
I . PRESS RETURN
J . PRINT
K . QUESTION MARK
L . SEMICOLON(S)
M . TURN ON THE CRT

Lesson Six Self-Check Test #2

Directions — If the statement is TRUE, write a "T" in the space provided. If the statement is FALSE, put an "F" in the space and change the italicized portion to make the statement TRUE.

——— 1. The number of statements which may be placed on a single line is limited only by the ***maximum*** size of the line (255 characters).

——— 2. When an IF-THEN statement is used on a multiple statement line, all commands following the "THEN" portion of the command will be processed only when the condition of the "IF" portion of the command is ***false***.

——— 3. A good technique when testing a program is to ***PRINT*** the program starting at a specific line number.

——— 4. The advantage of multiple statement lines is that it allows you to write ***shorter and faster*** running programs.

——— 5. The "bell" sound can be ***assigned to a variable*** for ease of use within a program.

——— 6. The command CHR$(7) can be used within a ***PRINT*** statement.

——— 7. The disadvantage of using multiple statements is that the programs are ***harder*** to debug and explain.

——— 8. The syntax for the DEL command when deleting several lines at a time is: DEL ***ending*** line #, ***beginning*** line #.

——— 9. Longer programs can be written as a series of ***sections*** put together.

——— 10. When outlining and structuring a program, the recommendations given in this course is to call the first section ***HOUSEKEEPING***.

——— 11. A good programming practice is to give the user some notification that the program has ***begun***.

Assignment For Lesson Six

Your assignment is to write a game program for two players. The game you will write is called "TWENTY-THREE MATCHES".

The game starts with a pile of 23 matches lying on a table. The two players alternate moves. Each player may take one, two or three matches from the pile. The player who takes the last match loses.

The assignment is divided into six small sections. Sections one through three are the steps needed to prepare the play of the game. Sections four and five control the play of the game. Section six is the ending routine.

Assignment Six Outline and Logic

Section #1: HOUSEKEEPING: Identify the program with REMarks and assign the bell sound to "BL$".

Section #2: INTRODUCTION: PRINT the title of the game in FLASHing. Use prompted INPUTs to accept the players' names and store then in "P1$" and "P2$". Assign the number of matches, 23, to the variable "ML".

Section #3: GAME INSTRUCTIONS: PRINT the instructions and use a dummy INPUT to hold the instructions on the screen until one of the players presses the RETURN key.

Section #4: PLAYER NUMBER ONE TAKES SOME MATCHES: Clear the screen and PRINT the number of matches left, variable "ML", in INVERSE. PRINT player one's name and instruct player one to take some matches. Use a prompted INPUT to allow the player to take one, two or three matches. Store the number taken in the variable "MT". Verify that the player took a legal number of matches, i.e., one, two or three.

Subtract the number of matches taken from the matches left with the formula: "ML = ML - MT". If any matches remain, GOTO the line number which begins player two's turn. Otherwise, sound the bell, PRINT player one loses, FLASHing, and GOTO the ending section.

Section #5: PLAYER TWO TAKES SOME MATCHES: Player two's turn is the same as player one's turn. The line numbers will be different, but the logic will be the same.

Section #6: END OF GAME: The ending section for "TWENTY-THREE MATCHES" is the same as the ending section for the "NUMBER-GUESS" game analyzed earlier in this lesson.

Additional Assignment For Lesson Six, #1

Write a program which allows the operator to do the following: enter a phrase to be printed; define the speed and mode in which to print the phrase.

Write the program according to the following program outline and logic paragraphs:

MODULE ONE: HOUSEKEEPING

Include program description (line 10) and SAVE command (line 20) in opening remarks.

MODULE TWO: INSTRUCTIONS

Begin in line 1000. Clear screen and instruct operator to enter phrase along with the print mode and SPEED at which the phrase will be displayed, or to enter the word "STOP" to end program.

Use dummy INPUT to hold instructions on CRT until operator has read them.

MODULE THREE: INPUT SECTION

Begin with line 2000. Prompt "ENTER THE PHRASE". Store phrase in "W$". Use IF-THEN to test W$ = "STOP". If true, GOTO end module. Prompt operator to enter print mode: "1 = NORMAL 2 = FLASH 3 = INVERSE". Store number of print mode in "M". Use IF-THEN with OR to test for "M" within range of 1 to 3. Prompt "CHOOSE SPEED FOR PRINT (0-255)". Store speed number in "S". Use IF-THEN with OR to test for "S" within range of 0 to 255.

MODULE FOUR: PRINT SECTION

Begin with line 3000. Clear screen. Use IF-THENs to set print mode based on value of "M". Use SPEED = command to set speed to value of "S". Print phrase on line 12, position 10. Return print mode to NORMAL. Set speed to 255. On line 20, position 1, prompt "(PRESS RETURN)" using dummy INPUT. GOTO beginning of "Input Module".

MODULE FIVE: END OF JOB SECTION

Begin with row 9000. Clear screen and end program.

Additional Assignment For Lesson Six, #2

Write a game program where you, the programmer, define a position on the screen where an invisible creature is hiding.

Allow the operator to enter row and column numbers until he/she finds the creature, or to enter zero to end the program.

Before writing the program, prepare a program outline of the program sections and write logic paragraphs for each section.

Give the operator adequate instructions for running the program before you begin to play.

LESSON SEVEN: *LOOPS*

LESSON OBJECTIVES

In Lesson Seven you are introduced to the concept of **FOR-NEXT** *loops.* To "loop" in a BASIC program simply means to repeat a series of program lines.

Through the use of the **FOR-NEXT** command pair, you can create a loop with a specified number of passes, or repetitions.

You are also introduced to the concept of an *"accumulator"* which refers to using a numeric variable to count the number of passes of a loop or to accumulate a column total.

You will learn that **FOR-NEXT** loops use a *"loop variable"*, which is a numeric variable whose value will increase from the beginning to the ending value of the loop. You will learn to use the **STEP** command to increase or decrease the loop variable by a value other than one.

You will learn a technique called *"nesting"* in which a loop is placed within another loop.

The new BASIC commands introduced in this lesson are: **FOR, NEXT** and **STEP.**

Program Listing For Lesson 7

GOTO Loop

```
900 I = 0
1000 I = I + 1
1010  PRINT I
1020  GOTO 1000
```

IF-THEN Loop

```
900 I = 0
1000 I = I + 1
1010  PRINT I
1020  IF I < 10 THEN 1000
```

FOR-NEXT Loop

```
1000  FOR I = 1 TO 10
1010  PRINT I
1020  NEXT I
```

Timing Loop Example: Version #1

```
10  REM  :: TIMING LOOP EXAMPLE
20  REM  SAVE TIMING LOOPS
1000  HOME
1010  INVERSE
1019  REM :: FIRST TIMED MESSAGE
1020  VTAB 4
1030  HTAB 12
1040  PRINT " ABOUT 1 SECOND "
1050  FOR I = 1 TO 740
1060  NEXT I
1070  HOME
1079  REM :: 2ND TIMED MESSAGE
1080  VTAB 12
1090  HTAB 12
1100  PRINT " ABOUT 5 SECONDS "
1110  FOR I = 1 TO 5 * 740
1120  NEXT I
1130  HOME
1139  REM :: 3RD TIMED MESSAGE
1140  VTAB 20
1150  HTAB 12
1160  PRINT " ABOUT 7 SECONDS "
1170  FOR I = 1 TO 7 * 740
1180  NEXT I
1190  HOME
1200  NORMAL
```

Timing Loop Example: Final Version

```
10  REM  :: TIMING LOOP EXAMPLE
20  REM  SAVE TIMING LOOPS
900 B = 1
910 E = 740
1000  HOME
1010  INVERSE
1019  REM :: FIRST TIMED MESSAGE
1020  VTAB 4
1030  HTAB 12
1040  PRINT " ABOUT 1 SECOND "
1050  FOR I = B TO E
1060  NEXT I
1070  HOME
1079  REM :: 2ND TIMED MESSAGE
1080  VTAB 12
1090  HTAB 12
1100  PRINT " ABOUT 5 SECONDS "
1110  FOR I = B TO 5 * E
1120  NEXT I
1130  HOME
1139  REM :: 3RD TIMED MESSAGE
1140  VTAB 20
1150  HTAB 12
1160  PRINT " ABOUT 7 SECONDS "
1170  FOR I = B TO 7 * E
1180  NEXT I
1190  HOME
1200  NORMAL
```

Stepping by Twos

```
10  REM :: STEPPING BY TWO
20  REM  SAVE TWO STEP
1000  HOME
1010  FOR I = 1 TO 23 STEP 2
1020  VTAB I
1030  PRINT "EVERY OTHER LINE"
1040  NEXT I
1050  END
```

Step Backwards

```
10  REM  :: STEP BACKWARDS
20  REM  SAVE BACKWARDS
1000  HOME
1010  SPEED= 150
1020  FOR I = 23 TO 1 STEP  - 1
1030  VTAB I
1040  PRINT "I CAN STEP BACKWARDS"
1050  NEXT I
1060  HOME
1070  SPEED= 255
1080  END
```

Correct Example Of Nested Loops

```
1000  FOR I = 1 TO 10
1010  FOR J = 1 TO 20
1020  NEXT J
1030  NEXT I
1040  PRINT "LOOP COMPLETED"
```

Incorrect Example Of Nested Loops

```
1000  FOR I = 1 TO 10
1010  FOR J = 1 TO 20
1020  NEXT I
1030  NEXT J
1040  PRINT "LOOP COMPLETED"
```

Nested Loops: Version #1

```
10  REM  :: NESTED LOOPS
20  REM  SAVE NESTED LOOPS
1010  FOR I = 1 TO 4
1060  FOR J = 1 TO 3
1080  NEXT J
1090  NEXT I
```

Nested Loops: Final Version

```
10  REM :: NESTED LOOPS
20  REM  SAVE NESTED LOOPS
500  SPEED= 100
1000  HOME
1010  FOR I = 1 TO 4
1020  VTAB I * 3: HTAB 1
1030  PRINT "FOR THE OUTSIDE LOOP PASS ";I
1040  HTAB 10
1050  PRINT "THE INSIDE LOOP CYCLES ";
1060  FOR J = 1 TO 3
1070  PRINT J;" ";
1080  NEXT J
1090  NEXT I
1100  SPEED= 255
9000  END
```

Lesson Seven Self-Check Test #1

Directions — Choose the term at the bottom of the page that BEST completes the statements. Some statements will require more than one answer. The space corresponds with the numbered blank. Write the letter of your choice in the space provided. The terms can be used more than once.

1. ——— The ——(1)—— command allows programmers to determine how many times a loop repeats itself.

2. ——— A variable which counts the number of times a loop repeats itself is called a(n) ——(2)——.

3. ——— The ——(3)—— loop serves both the function of counting the number of times a loop repeats and the function of the ——(4)—— command.

4. ———

5. ——— When "1" is added to the controlling variable for each pass of the loop, it is called ——(5)—— by one.

6. ——— To print every other line, the command ——(6)—— can be used following the FOR statement.

7. ——— The technique of placing a loop within a loop is called ——(7)——.

8. ——— The ——(8)—— is a variable whose value increases with each pass of a loop. It is basically a counter.

9. ——— A ——(9)—— loop can create a timing loop to control the length of time a printed message is displayed.

10. ——— Two or more loops can be nested in a program as long as the ——(10)—— is completely contained within the ——(11)——.

11. ———

12. ——— The computer can be instructed to ——(12)—— backwards.

13. ——— In graphics, variables are used frequently to determine the beginning and ending value of the ——(13)—— loop.

A . ACCUMULATOR
B . FOR-NEXT
C . IF-THEN
D . INSIDE LOOP
E . NESTING
F . OUTSIDE LOOP
G . PRINTING
H . STEP
I . STEPPING

Lesson Seven Self-Check Test #2

Directions — Carefully read the statement and each of the choices. Only one choice will BEST fit the statement. Place the letter of your choice in the space provided.

——— 1. These programming lines:

```
900 I = 0
1000 I=I+1
1010 PRINT
1020 IF I<10 THEN 1000
```

are an example of a

A. LOOP
B. STEP
C. NEST
D. NEXT

——— 2. These programming lines:

```
900 I = 0
1000 I=I + 2
1010 PRINT I
1020 GOTO 1000
```

would be an example of a(n)

A. Print Statement
B. Accumulating Variable
C. Eternal Loop
D. Calculating Section

——— 3. In this command line:

FOR I=1 to 25

"I" is equal to

A. 26 because APPLE II is a base zero machine.
B. 1 up to 25 and no more.
C. groups of 1 to 25 depending on how many loops are performed.
D. 24 because it begins on 1 and ends on 25.

——— 4. To establish a timing loop of approximately one second, the loop should begin with:

FOR I=

A. 1 to 714
B. 1 to 214
C. 1 to 1010
D. 1 to 740

——— 5. This line:

FOR I = 1 to 23 STEP2

would cause a loop to be repeated

A. 11 times.
B. 10 times.
C. 12 times.
D. 9 times.

——— 6. When variables are used to determine the beginning and ending value of a FOR-NEXT loop, they are assigned their value

A. just before the FOR-NEXT loop.
B. before the main program section.
C. just after the FOR-NEXT loop.
D. just after the timing loop.
E. anywhere before the FOR-NEXT loop.

——— 7. The term accumulator does not describe a special kind of variable. It merely describes a special

A. calculation of a variable.
B. acronym of a variable.
C. variable name.
D. use of a variable.

——— 8. The FOR-NEXT pair provides us with a simplified means of

A. creating a way to branch in a program.
B . creating and controlling multiple statement lines.
C . creating and controlling a loop in a program.
D . creating and controlling listing in a program.

Assignment #1 For Lesson Seven

Your first assignment for this lesson is to write a "Self-Destruct" program. The program is to count down the seconds from ten to one, and then PRINT the word "BOOM" in FLASH mode in three columns on the screen.

To count down the seconds, the program should use a timing loop whose beginning value is ten and whose ending value is zero with a STEP minus one.

The program should PRINT the number of seconds remaining and should instruct the computer to make its bell sound once for each second.

Remember, a timing loop with 740 passes takes about one second.

When the timer reaches zero, the program should use another FOR-NEXT loop to PRINT the word "BOOM" in three columns on each screen line in FLASH mode.

Assignment #2 For Lesson Seven

The second assignment is to write a program which PRINTs an asterisk (*) in every other column, on every other line on the screen.

The program should use a double nested loop (two loops, one inside the other).

The value of the outside loop variable controls the vertical positioning of the cursor to the appropriate screen line. The value of the inside loop variable controls the horizontal positioning of the cursor to the appropriate column on each line.

Use VTAB and HTAB commands to position the cursor. After each asterisk is PRINTed, use a semicolon to instruct the computer to PRINT the next asterisk on the same line. The VTAB command will instruct the computer to move to the next screen line. The outside loop variable will be used with the VTAB command and the inside loop variable will be used with the HTAB command.

You may use the STEP command to instruct the computer to skip lines and columns, or you may use a calculation to instruct the computer to skip lines and columns (for example: VTAB I*2 or HTAB I*2).

Additional Assignment For Lesson Seven, #1

Write a program which uses two accumulators. Start the first accumulator at zero and the second at one hundred.

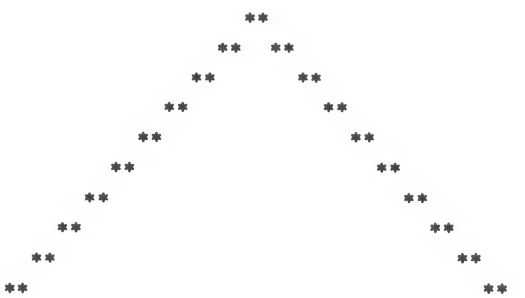
Create a GOTO loop in which you add one to the first accumulator, subtract one from the second, and print both accumulators.

Use an IF-THEN to stop the program when both the accumulators are equal.

The program would not have worked if the second accumulator started at 99. Change the program to start the second accumulator at 99 to find out why.

Additional Assignment For Lesson Seven, #2

Write a program which prints the following on the screen twenty times.



Start printing on screen line nineteen with the lower left tip of the arrow. PRINT up to the point of the arrow and back down to the lower right tip. To make the arrow scroll up the screen, you will need to PRINT a blank space on line twenty-four. FOR-NEXT loops must be used to control the printing.

LESSON EIGHT: *SUBROUTINES*

LESSON OBJECTIVES

In Lesson Eight you will be introduced to *subroutines, branching* and *flowcharting.*

A *subroutine* is a set of programming lines which may be accessed from any portion of a program.

The term *branching* is derived from the branches of a tree. Just as a tree has a single trunk which splits into several branches, a BASIC program may have one main program module which can then execute one of several program tasks, or "branches".

Flowcharting is another technique used when designing and analyzing programs. Flowcharts use symbols to identify the operations in a program, and lines to illustrate the program "flow".

The new BASIC commands introduced in this lesson are: **GOSUB, RETURN, ON variable GOTO and ON variable GOSUB.**

Program Listing For Lesson 8

Sub Control

```
1000  REM .. THE MAIN PGM MODULE
1010  REM .. CONTROLS SUBROUTINES
2000  GOSUB 6000
2010  REM  ...
3000  REM  ...
3010  REM  ...
4000  GOTO 9000
6000  REM  SUBROUTINE
6010  REM  ...
6500  RETURN
9000  END
```

Quiz

```
10  REM :: MULTIPLE CHOICE PROGRAM
20  REM  SAVE QUIZ
1000  HOME
1010  PRINT "IN A NESTED LOOP THE INNER LOOP MUST "
1020  PRINT "    1 - BE INSIDE THE OUTER LOOP"
1030  PRINT "    2 - EXECUTE FIVE TIMES"
1040  INPUT "ENTER NUMBER OF CHOICE ";A
1050  IF A < 1 OR A > 2 THEN 1000
1060  IF A = 1 THEN  GOSUB 7000
1070  IF A = 2 THEN  GOSUB 8000
2000  HOME
2010  PRINT "A CHARACTER VARIABLE MUST END WITH"
2020  PRINT "    1 - PERCENT SYMBOL"
2030  PRINT "    2 - DOLLAR SIGN"
2040  INPUT "ENTER NUMBER OF CHOICE ";B
2050  IF B < 1 OR B > 2 THEN 2000
2060  IF B = 1 THEN  GOSUB 8000
```

```
2070  IF B = 2 THEN  GOSUB 7000
2090  GOTO 9000
6999  REM :: CORRECT ANSWER ROUTINE
7000  VTAB 22
7010  PRINT "THAT IS CORRECT"
7020  FOR I = 1 TO 3 * 740
7030  NEXT I
7040  RETURN
7999  REM :: INCORRECT ANSWER ROUTINE
8000  VTAB 22
8010  PRINT "SORRY, THAT IS INCORRECT"
8020  FOR I = 1 TO 3 * 740
8030  NEXT I
8040  RETURN
9000  HOME
9010  PRINT "THE PROGRAM HAS ENDED"
9020  END
```

Sub Control: Error 1

```
1000  REM .. THE MAIN PGM MODULE
1010  REM .. CONTROLS SUBROUTINES
2000  GOTO 6000
2010  REM ...
3000  REM ...
3010  REM ...
4000  GOTO 9000
6000  REM SUBROUTINE
6010  REM ...
6500  RETURN
9000  END
```

Sub Control: Error 2

```
1000  REM .. THE MAIN PGM MODULE
1010  REM .. CONTROLS SUBROUTINES
2000  GOSUB 6000
2010  REM ...
3000  REM ...
3010  REM ...
6000  REM SUBROUTINE
6010  REM ...
6500  RETURN
9000  END
```

Dictionary

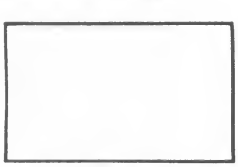
```
10  REM : DICTIONARY PROGRAM
20  REM SAVE DICTIONARY
1000  HOME
1010  PRINT  TAB( 11)"DICTIONARY PROGRAM"
1020  PRINT
1030  PRINT "WHICH DEFINITION DO YOU WANT TO SEE"
1040  PRINT
1050  PRINT "1 - MICROCOMPUTER"
```

```
1060  PRINT "2 - HIGH LEVEL LANGUAGE"
1070  PRINT "3 - ABACUS"
1080  PRINT
1090  INPUT "ENTER NUMBER OF CHOICE OR ZERO TO END ";A
1100  IF A = 0 THEN 9000
1110  IF A < 1 OR A > 3 THEN 1000
1120  HOME
1130  ON A GOSUB 2000,3000,4000
1140  FOR I = 1 TO 5 * 740
1150  NEXT I
1160  GOTO 1000
2000  PRINT "A MICROCOMPUTER IS A VERY SMALL COMPUTER"
2010  PRINT "CONTROLLED BY A TINY SILICON CHIP"
2020  RETURN
3000  PRINT "A HIGH LEVEL LANGUAGE IS A PROGRAMMING "
3010  PRINT "LANGUAGE WHOSE COMMANDS ARE CLOSE TO"
3020  PRINT "ENGLISH WORDS"
3030  RETURN
4000  PRINT "AN ABACUS IS AN ANCIENT DEVICE USED FOR"
4010  PRINT "MATHEMATICAL CALCULATIONS"
4020  RETURN
9000  HOME
9010  PRINT "THE PROGRAM HAS ENDED"
9020  END
```

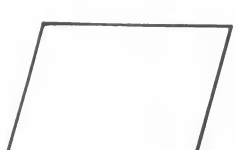
ON GOTO

```
10  REM  : ON GOTO PROGRAM
20  REM  SAVE ON GOTO
1000  HOME
1010  INPUT "ENTER A NUMBER (1-4) OR ZERO TO END ";N
1020  IF N = 0 THEN 9000
1030  ON N GOTO 2000,3000,4000,5000
1040  INPUT "PRESS RETURN ";A$
1050  GOTO 1000
2000  PRINT "LINE 2000"
2010  GOTO 1040
3000  PRINT "LINE 3000"
3010  GOTO 1040
4000  PRINT "LINE 4000"
4010  GOTO 1040
5000  PRINT "LINE 5000"
5010  GOTO 1040
9000  HOME
9010  END
```

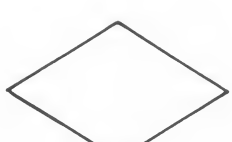
Flowcharting



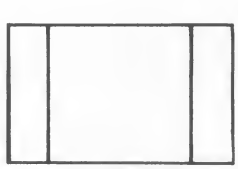
Process Block



Input/Output

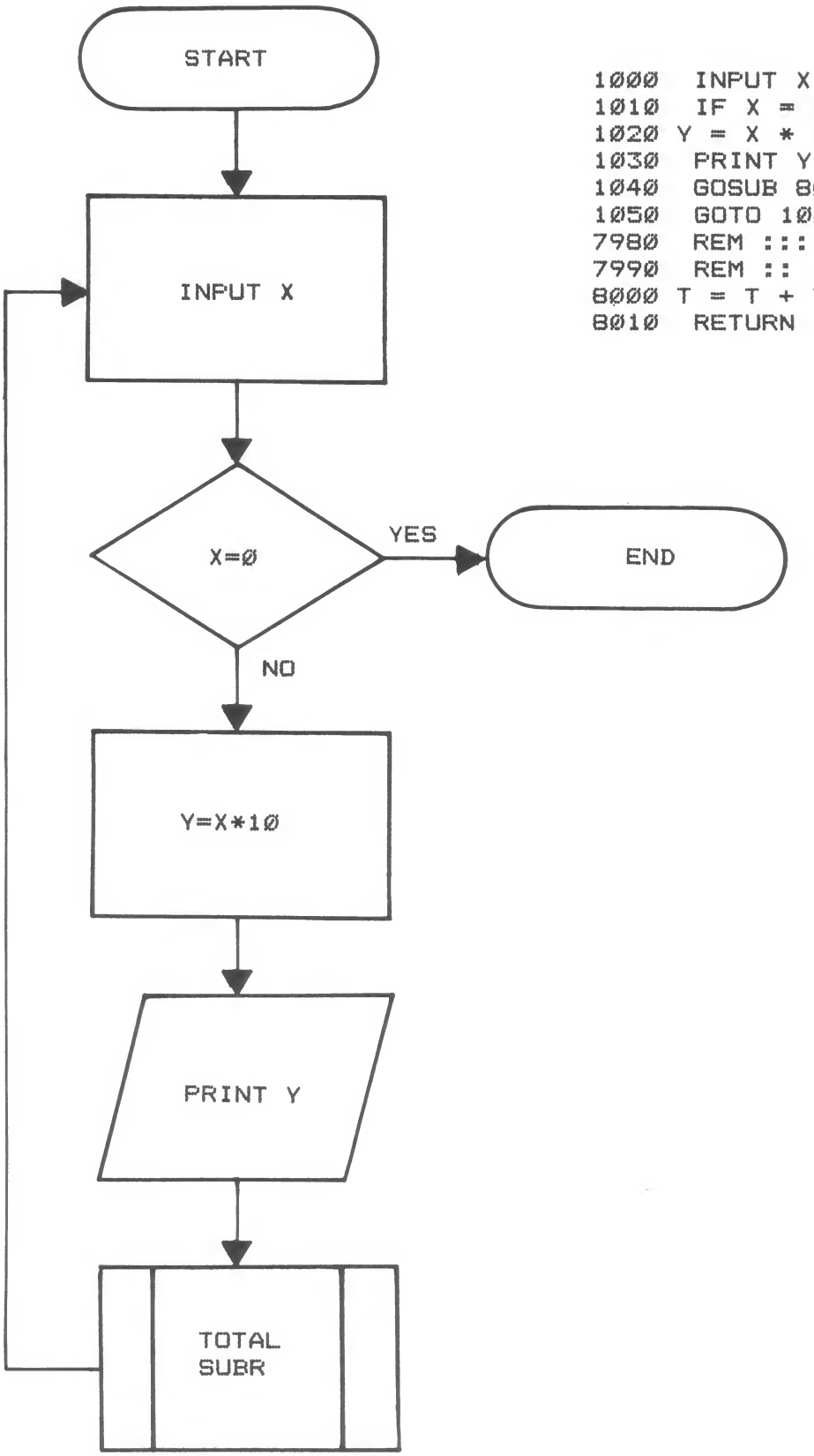


Decision



Predefined Process

Terminal




```
1ØØØ  INPUT X
1Ø1Ø  IF X = Ø THEN  END
1Ø2Ø Y = X * 1Ø
1Ø3Ø  PRINT Y
1Ø4Ø  GOSUB 8ØØØ
1Ø5Ø  GOTO 1ØØØ
7981  REM ::::::::::::::::
7991  REM ::  TOTAL SUBR
8ØØØ T = T + Y
8Ø1Ø  RETURN
```

Lesson Eight Self-Check Test #1

Directions — Fill in the blanks with the correct command, term, or name.

1.———————————— A ——(1)—— is a series of program lines which performs a specific task and ends with the command RETURN.

2.———————————— GOTO and ——(2)—— are not interchangeable.

3.———————————— A programmer must be very careful that the program does not ——(3)—— into a subroutine.

4.———————————— ——(4)—— is frequently called a calculated, or computed GOTO.

5.———————————— The traditional method of program design using symbols and lines is called ——(5)——.

6.———————————— The tool used by programmers for program design using symbols is called a ——(6)——.

7.———————————— There is only one proper way to EXIT from a subroutine — with a ——(7)—— command.

8.———————————— ——(8)—— lines within a program will help identify routines for the programmer.

9.———————————— Organizing a program into a ——(9) ——and a series of subroutines saves a programmer many hours of debugging and testing time.

10.———————————— The ——(10)—— symbol is used when designing programs to indicate the beginning and ending of programs.

11.———————————— A diamond shape in program design would indicate a ——(11)—— to the programmer.

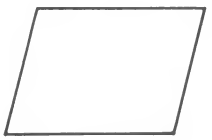
12.———————————— The paired commands ——(12)—— and ——(13)—— are used for subroutines.

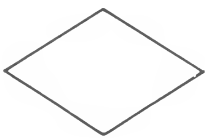
13.————————————

Lesson Eight Self-Check Test #2

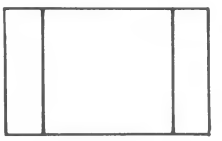
Directions — Each item of the Program Design Symbols in the key are used by programmers in flowcharting. Below the key is a list of uses and definitions. In the space provided, select the symbol that goes with the statement.

A .

B . 

C . 

D .

E . 

——— 1. A DECISION

——— 2. SUBROUTINE END

——— 3. GOSUB USUALLY ASSOCIATED WITH THIS SYMBOL

——— 4. REPRESENTS PREDEFINED PROCESS

——— 5. BEGINS PROGRAMS

——— 6. MOST GENERAL OF THE SYMBOLS

——— 7. USED WITH THE IF-THEN

——— 8. ENDS PROGRAM

——— 9. INPUT GOES WITH THIS SYMBOL

——— 10. USED IN PLACE OF A SERIES OF PROGRAM LINES WHICH PERFORMS A SINGLE TASK

——— 11. TERMINAL SYMBOL

——— 12. PRINT GOES WITH THIS SYMBOL

——— 13. USED WITH FOR-NEXT

——— 14. ANY OPERATION NOT CORRESPONDING WITH ONE OF THE OTHER FOUR SYMBOLS

Lesson Eight Self-Check Test #3

Directions — Choose the answer that BEST completes the statement below. Write the letter of the choice in the space provided.

——— 1. The main program module directs

A. at what point the subroutine is to return back to the GOSUB.
B. where the subroutine must go to be performed.
C. the subroutines, GOSUB and RETURN.
D. the order in which subroutines perform their designated tasks.

——— 2. If the same series of operations is used in many different places within a program it can

A. be dropped from the program after three uses.
B. be placed in a subroutine to save programming lines.
C. be placed in a GOTO command line and referred to by line number.
D. be placed in several subroutines and used when necessary.

——— 3. The command that instructs the computer to process commands in a subroutine beginning at a specific line number is

A. GOTO
B. GO-ON
C. GOSUB
D. ON-GOTO

——— 4. In the statement:

1050 ON A GOSUB 1100,1200,1300

If A=2, the subroutine to be executed will begin on

A. LINE 1050
B. LINE 1100
C. LINE 1200
D. LINE 1300

——— 5. Computed GOTOs are most commonly used in a program in which the operator

A. is given a choice between a number of operations.
B. is ready to use the basic math operations for computation.
C. must calculate the line number for the GOTO.
D. does not know which subroutine to execute next.

Assignment For Lesson Eight

Your assignment for this lesson is to write a program that acts as a four function calculator that adds, subtracts, multiplies, and divides.

First, allow the operator to enter two numbers. If both of the numbers are zeros, GOTO the ending section of the program. Otherwise, display the four calculator choices available and allow the operator to select one of the four calculations.

Then, clear the screen and use a "computed GOSUB" to execute one of four subroutines: addition subroutine, subtraction subroutine, multiplication subroutine and division subroutine.

Each of the calculating subroutines should first use the two numbers entered by the operator in the appropriate calculation. Then PRINT the equation and the results of the calculation on the CRT. For example, "1 + 2 = 3".

Finally, use a "dummy INPUT" to leave the equation on the CRT until the operator presses the RETURN key, and return to the top of the main program module to allow the operator to enter two more numbers or end the program.

Additional Assignment For Lesson Eight, #1

The following program has FIVE errors. Your assignment is to debug the program.

```
10  REM :: LESSON 8 ADDITIONAL 1
20  REM  SAVE LES8A1
1000  GOTO 8000
1010  PRINT "PLEASE SELECT..."
1020  PRINT " 1 - NAME...: ";N$
1030  PRINT " 2 - ADDRESS: ";A$
1040  PRINT " 3 - CITY...: ";C$
1050  PRINT
1060  NORMAL
1070  PRINT "ENTER 1 - 3 OR 0 TO END: ";S
1080  IF S = 0 THEN 5000
1090  AT S GOTO 1200,2000,3000
1100  GOTO 1070
1200  GOSUB 8000
1210  PRINT "CURRENT NAME IS ";N$
1220  INPUT "ENTER CORRECT NAME: ";N$
1230  GOTO 1000
2000  GOSUB 8000
2010  PRINT "CURRENT ADDRESS IS ";A$
2020  INPUT "ENTER CORRECT ADDRESS: ";A$
2030  GOTO 1000
3000  GOSUB 8000
3010  PRINT "CURRENT CITY IS ";C$
3020  INPUT "ENTER CORRECT CITY: ";C$
3030  GOTO 1000
5000  GOSUB 8000
5010  PRINT "PLEASE VERIFY ENTRIES"
5020  PRINT " NAME...: ";N$
5030  PRINT " ADDRESS: ";A$
```

```
5040  PRINT " CITY...: ";C$
5050  PRINT
5060  INPUT "ENTER Y OR N: ";AN$
5070  IF AN$ = "N" THEN 1000
8000  HOME
8010  VTAB 8
8020  INVERSE
8030  RETURN
9000  GOSUB 8000
9010  PRINT "PROGRAM ENDED"
9020  END
```

Additional Assignment For Lesson Eight, #2

Write a program which allows the operator to enter any word or "STOP" to end the program. Tell the operator that the word can have a length of one to five characters.

Use four subroutines to:
1) print the word at several positions from left to right.
2) print the word at several positions from right to left.
3) print the word on 23 lines from top to bottom.
4) print the word on 23 lines from bottom to top.

Use an ON-GOSUB within a FOR-NEXT loop to execute the four subroutines in order.

Allow the operator to continue entering words until the word "STOP" is entered. Then end the program.

LESSON NINE: *ARRAYS*

LESSON OBJECTIVES

In Lesson Nine you will be introduced to the concept of *arrays*. You will learn that an array is a set of variables linked together by a common name, and that each variable of the set is called an *element*. You will learn to reference an array element by its number, called a *subscript*.

You will learn that the APPLE is a *"base zero"* computer which simply means that the first subscript of an array is zero, instead of one.

You will also learn how to assign values to variables using the command pair **DATA** and **READ.**

The new BASIC commands introduced in this lesson are: **DIM, DATA** and **READ.**

Program Listing For Lesson 9

Example Dimension Statements

```
50  DIM M$(11)

50  DIM A$(15),B(11)
```

Arrays

```
10  REM  ARRAY USAGE
20  REM  SAVE NAME ARRAYS
30  REM  HOUSEKEEPING
100  DIM NA$(4),AD$(4)
990  REM  INPUT SECTION
1000  FOR I = 0 TO 4
1010  HOME
1020  VTAB 5
1030  INPUT "ENTER A NAME ... ";NA$(I)
1040  VTAB 9
1050  INPUT "ENTER AN ADDRESS ... ";AD$(I)
1060  NEXT I
1990  REM  PRINT SECTION
2000  HOME
2010  FOR I = 0 TO 4
2020  PRINT NA$(I)
2030  PRINT  TAB( 5);AD$(I)
2040  NEXT I
9000  END
```

DATA/READ Example

```
100  DATA  JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY,AUGUST,
     SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
200  READ M1$,M2$,M3$,M4$,M5$,M6$,M7$,M8$,M9$,MA$,MB$,MC$
```

DATA 1

```
10  REM :: DATA AND READ PROGRAM ::
20  REM SAVE DATA 1
100  DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY
110  DATA AUGUST,SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
600  READ M1$,M2$,M3$,M4$,M5$,M6$,M7$,M8$,M9$,MA$,MB$,MC$
999  REM :: BEGIN PROGRAM HERE ::
1000  HOME
1010  VTAB 5
1020  INPUT "ENTER A MONTH NUMBER FROM 1 TO 12: ";N
1030  IF N = 0 GOTO 5000
1040  ON N GOTO 1100,1200,1300,1400,1500,1600,1700,1800,1900,
      2000,2100,2200
1050  GOTO 1000
1099  REM :: BRANCHING GOTO PRINT SECTIONS
1100  HOME
1110  VTAB 10
1120  PRINT "THE MONTH IS ";M1$
1130  GOTO 1010
1200  HOME
1210  VTAB 10
1220  PRINT "THE MONTH IS ";M2$
1230  GOTO 1010
1300  HOME
1310  VTAB 10
1320  PRINT "THE MONTH IS ";M3$
1330  GOTO 1010
1400  HOME
1410  VTAB 10
1420  PRINT "THE MONTH IS ";M4$
1430  GOTO 1010
1500  HOME
1510  VTAB 10
1520  PRINT "THE MONTH IS ";M5$
1530  GOTO 1010
1600  HOME
1610  VTAB 10
1620  PRINT "THE MONTH IS ";M6$
1630  GOTO 1010
1700  HOME
1710  VTAB 10
1720  PRINT "THE MONTH IS ";M7$
1730  GOTO 1010
1800  HOME
1810  VTAB 10
1820  PRINT "THE MONTH IS ";M8$
1830  GOTO 1010
1900  HOME
1910  VTAB 10
1920  PRINT "THE MONTH IS ";M9$
1930  GOTO 1010
2000  HOME
2010  VTAB 10
2020  PRINT "THE MONTH IS ";MA$
2030  GOTO 1010
2100  HOME
2110  VTAB 10
```

```
2120  PRINT "THE MONTH IS ";MB$
2130  GOTO 1010
2200  HOME
2210  VTAB 10
2220  PRINT "THE MONTH IS ";MC$
2230  GOTO 1010
5000  HOME
5010  FOR I = 1 TO 15
5020  PRINT  TAB( I * 2);"END OF JOB"
5030  NEXT I
5040  END
```

Months

```
10  REM ::MONTH NAME ASSIGNMENT ::
20  REM SAVE MONTHS
100  DIM M$(12)
500  DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY,AUGUST,
     SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
600  FOR I = 1 TO 12
610  READ M$(I)
620  NEXT I
999  REM :: BEGIN PROGRAM HERE ::
1000  HOME
1010  VTAB 5
1020  INPUT "ENTER THE NUMBER OF THE MONTH ";N
1030  IF N = 0 GOTO 5000
1040  IF N < 1 OR N > 12 THEN 1000
1050  HOME
1060  VTAB 9
1070  PRINT "THE NAME OF THE MONTH IS ";M$(N)
1080  GOTO 1010
5000  HOME
5010  FOR I = 1 TO 15
5020  PRINT  TAB( I * 2);"END OF JOB"
5030  NEXT I
5040  END
```

Multi-Dimensional Arrays

```
10  REM :: MULTI-DIMENSIONAL ARRAYS ::
20  REM SAVE MULTI ARRAYS
99  REM DIMENSION A TWO DIMENSIONAL ARRAY
100  DIM NA$(3,3)
990  REM :: ARRAY INPUT SECTION
1000  FOR I = 0 TO 3
1010  HOME
1020  VTAB 5
1030  INPUT "ENTER A NAME .... ";NA$(I,0)
1040  VTAB 9
1050  INPUT "ENTER AN ADDRESS: ";NA$(I,1)
1060  VTAB 13
1070  INPUT "ENTER A CITY ..... ";NA$(I,2)
1080  VTAB 17
1090  INPUT "ENTER ZIP CODE... ";NA$(I,3)
```

```
1100  NEXT I
1990  REM :: ARRAY PRINT SECTION
2000  HOME
2010  FOR I = 0 TO 3
2020  FOR J = 0 TO 3
2030  PRINT NA$(I,J)
2040  NEXT J
2050  NEXT I
2060  END
```

Lesson Nine Self-Check Test #1

Directions — If the statement is TRUE, place a "T" in the space provided. If the statement is FALSE, place an "F" in the space and change the italicized portion to make the statement TRUE.

——— 1. An array is a set of variables that are linked together by a common ***number.***

——— 2. A part of an array is called an ***element***.

——— 3. Each element in an array is ***a group***.

——— 4. Numbering an element in an array is called ***subcursing***.

——— 5. Applesoft BASIC is a ***base zero*** language and therefore begins subscripting at zero.

——— 6. We can dimension two different arrays on the same line by separating the array names with a ***semicolon***.

——— 7. It ***is not necessary*** to put data within a program on one statement line.

——— 8. Using the ***FOR-NEXT*** commands is the only case in which strings can be assigned to variables without quotation marks.

——— 9. By using a ***multi-dimensional array*** in a program, information can be put together in a logical group.

——— 10. It is possible to have more than ***two*** dimensions in an array.

——— 11. The computer recognizes the first ***three*** characters of a variable name.

——— 12. A ***loop*** can be created to enter data and to store it in an array.

——— 13. A dimension statement for a 12 element array could be written: ***50 DIM M$(11)***

——— 14. The computer will automatically dimension an array with ***five*** elements if no DIM statement is used.

Lesson Nine Self-Check Test #2

Directions — Match the term at the bottom of the page with the definitions or statements below by writing the letter of your choice in the space provided. Terms may be used more than once. Please note there are more terms than definitions or statements.

——— 1. A set of these are linked together to form an array.

——— 2. A computer must know how many of these are in an array.

——— 3. Numbering an element in an array is called ———.

——— 4. This command tells the computer what the largest subscript will be.

——— 5. If you are using two arrays and wish to coordinate them, this command will create a loop to do that.

——— 6. If an operator places this in an INPUT statement without putting the statement in quotes, the computer will ignore anything typed after it.

——— 7. Another way to assign data to variables besides LET and INPUT is with these paired commands.

——— 8. All this statement does is list information which will be placed in variables.

——— 9. To read data into individual elements in an array, this type of loop is used.

——— 10. This command actually assigns data into variables.

——— 11. This is a set of single dimensional arrays linked together.

——— 12. If an array is not dimensioned, the computer will dimension it with this number of elements.

——— 13. In order to access each element in a multi-dimensional array, this has to be used.

——— 14. In this statement: NA$(2,3) this number will instruct the computer to find the third single dimensioned array.

A . COMMA
B . DATA
C . DATA-READ
D . DIM
E . DOUBLE-NESTED
F . EIGHT
G . ELEMENTS
H . ELEVEN
I . FOR-NEXT
J . GOTO-ON
K . MULTI-DIMENSIONAL ARRAY
L . READ
M . SEMICOLON
N . SUBSCRIPTING
O . TEN
P . THREE
Q . TWELVE
R . TWO
S . VARIABLE

Assignment For Lesson Nine

Your assignment for this lesson is to write a program which allows the operator to enter the won, lost and tied records for four baseball teams.

The team names should be stored in a single dimensional character array. The records for the teams should be stored in a multi-dimensional numeric array.

In the "housekeeping section" of the program, first dimension a character array with four elements. Then dimension a numeric array with four rows each with three elements: DIM T$(3), T(3,2).

Place the names of the teams in a DATA statement and use a FOR-NEXT loop to read the team names into the character array.

In the "input" section" of the program, the operator is to enter the games won, lost and tied for each team.

First display the name of the team. Then prompt the operator to enter the team's record. The team records should be stored as follows: for team one the records will be stored in the "zero row" of the array; games won should be stored in the zero, zero element of the numeric array "T(0,0)"; games lost should be stored in the zero, one element "T(0,1)"; games tied should be stored in the zero, two element "T(0,2)".

The record for team two will be stored in the "one" row of the array. Games won will be stored in the "T(1,0)" array element; games lost in the "T(1,1)" element; and games tied in the "T(1,2)" element.

The record for team three will be stored in row two elements, and the record for team four will be stored in the row three elements.

After the records have been entered, write the "printing section" of the program as follows:

Clear the screen and PRINT the name of each team and team records using a nested loop.

The outside loop should begin "FOR I=0 TO 3". The outside loop will control the printing of the team names and will control which row of the multi-dimensional array will be printed.

Before beginning the inside loop, PRINT the team name.

The inside loop should begin "FOR J = 0 TO 2". The inside loop controls which element of the row is being printed.

When the printing is complete, the team names and records should be printed in rows and columns on the screen in a form similar to the following:

TEAM NAME	GAMES	WON	LOST	TIED
TEAM ONE		##	##	##
TEAM TWO		##	##	##
TEAM THREE		##	##	##
TEAM FOUR		##	##	##

Additional Assignment For Lesson Nine, #1

Write a program to help a baseball coach pick his team.

First dimension a character array with two rows, each having nine elements.

Store the name of each position in the elements of the first row of the two-dimensional array.

(The nine positions are pitcher, first base, second base, third base, shortstop, left field, center field, right field, and catcher.)

Prompt the coach to enter the names of the player for each of the positions one by one.

Store the players' names in the elements of the second row of the two-dimensional array.

When all of the positions have had a player assigned, print the players' names and positions on the CRT.

You must use the command pairs DATA-READ and FOR-NEXT in your answer.

Additional Assignment For Lesson Nine, #2

Write a program which allows an operator to enter up to twenty grades to be averaged.

Store the grades in an array as they are entered. Then print the average.

Averaging a series of grades requires two steps: (1) the grades must be added together; (2) the total of the grades must be divided by the number of grades. For example, to average the grades 80, 95, and 92, you first add the grades together ($80+95+92=267$). Then you divide the total by the number of grades ($267/3=89$). The average of the three grades, 80, 95, and 92, is 89.

LESSON TEN: *STRING FUNCTIONS*

LESSON OBJECTIVES

In Lesson Ten you will be introduced to the BASIC commands which allow you to extract single characters or portions of a string of characters. You will also be taught the command which allows you to put strings together to form new strings called *concatenation*, and the commands which convert numbers to strings and strings to numbers. These BASIC commands are called *"string functions"*.

In this lesson you will also learn the concept of a *null string* which is an empty string, a string with no value. (A null string has a length of zero.)

The new BASIC commands which are introduced in this lesson are: **LEN( ), LEFT$( ), RIGHT$( ), MID$( ), VAL( ),** and **STR$( ).** You are also introduced to the use of the *plus (+) sign* to put strings together.

Program Listing For Lesson 10

LEN Function

```
10  REM  * A PROGRAM ABOUT "LEN" *
20  REM  SAVE LEN-FUNCTION
1000  HOME
1010  INPUT "ENTER A WORD (/ TO END) ";W$
1020  IF W$ = "/" THEN 9000
1990  REM  * DISPLAY THE WORDS *
2000  VTAB 5
2010 L =  LEN (W$)
2020  PRINT W$" -- HAS ";L;" CHARACTERS."
2030  VTAB 10
2040  INPUT "PRESS RETURN FOR NEXT WORD ";A$
2050  GOTO 1000
9000  END
```

VAL Function

```
10  REM :: VAL FUNCTION
20  REM  SAVE VAL FUNCTION
1000 C$ = "150 COOKIES"
1010 N$ =  LEFT$ (C$,3)
1020 V =  VAL (N$)
1030  PRINT V * 25;" COOKIES"
1040  END
```

Date Separator

```
10  REM :: DATE SEPARATOR ::
20  REM  SAVE DATE SEPARATE
50  DIM M$(12)
60  DATA JAN,FEB,MAR,APR,MAY,JUN,JUL
70  DATA AUG,SEPT,OCT,NOV,DEC
80  FOR I = 1 TO 12
90  READ M$(I)
100  NEXT I
```

```
1000  HOME
1010  VTAB 8
1020  PRINT "DATE SEPARATOR"
1030  VTAB 12
1035  PRINT "ENTER DATE (MMDDYYYY)"
1040  INPUT "OR Ø TO END: ";D$
1045  IF D$ = "Ø" THEN 9000
1050  IF  LEN (D$) = 8 THEN 2000
1060  HOME : VTAB 14
1070  PRINT "IMPROPER DATE": GOTO 1010
1990  REM :: SEPARATE DATE ::
2000 MO$ =  LEFT$ (D$,2)
2010 DY$ =  MID$ (D$,3,2)
2020 YR$ =  RIGHT$ (D$,4)
2090  REM :: GET THE MONTH::
2100 M =  VAL (MO$)
2110 MN$ = M$(M)
2120 FM$ = MN$ + " " + DY$ + ", " + YR$
2130  PRINT "THE DATE YOU ENTERED WAS:"
2140  PRINT FM$
2150  GOTO 1010
9000  HOME : END
```

String Loops: Version #1

```
10  REM :: STRING MANIPULATIONS
20  REM  SAVE STRING LOOPS
500  HOME
1000  PRINT "ENTER A WORD OR PHRASE"
1010  INPUT "OR PRESS RETURN TO END: ";N$
1020  IF N$ = "" THEN 9000
1030  SPEED= 50
1040  HOME
1060  FOR I = 1 TO  LEN (N$)
1100  PRINT  MID$ (N$,I,1)
1120  NEXT I
1130  SPEED= 255
1140  PRINT
1150  GOTO 1000
9000  END
```

String Loops: Version #2

```
10  REM :: STRING MANIPULATIONS
20  REM  SAVE STRING LOOPS
500  HOME
1000  PRINT "ENTER A WORD OR PHRASE"
1010  INPUT "OR PRESS RETURN TO END: ";N$
1020  IF N$ = "" THEN 9000
1030  SPEED= 50
1040  HOME
1060  FOR I =  LEN (N$) TO 1 STEP  - 1
1100  PRINT  MID$ (N$,I,1)
1120  NEXT I
1130  SPEED= 255
1140  PRINT
1150  GOTO 1000
9000  END
```

String Loops: Version #3

```
10  REM :: STRING MANIPULATIONS
20  REM  SAVE STRING LOOPS
500  HOME
1000  PRINT "ENTER A WORD OR PHRASE"
1010  INPUT "OR PRESS RETURN TO END: ";N$
1020  IF N$ = "" THEN 9000
1030  SPEED= 50
1040  HOME
1050 J = 0
1060  FOR I =  LEN (N$) TO 1 STEP  - 1
1070 J = J + 1
1080  PRINT  MID$ (N$,J,1);
1100  PRINT  MID$ (N$,I,1)
1120  NEXT I
1130  SPEED= 255
1140  PRINT
1150  GOTO 1000
9000  END
```

String Loops: Final Version

```
10  REM :: STRING MANIPULATIONS
20  REM  SAVE STRING LOOPS
500  HOME
1000  PRINT "ENTER A WORD OR PHRASE"
1010  INPUT "OR PRESS RETURN TO END: ";N$
1020  IF N$ = "" THEN 9000
1030  SPEED= 50
1040  HOME
1050 J = 0
1060  FOR I =  LEN (N$) TO 1 STEP  - 1
1070 J = J + 1
1075  HTAB J
1080  PRINT  MID$ (N$,J,1);
1090  HTAB I
1100  PRINT  MID$ (N$,I,1)
1120  NEXT I
1130  SPEED= 255
1140  PRINT
1150  GOTO 1000
9000  END
```

Lesson Ten Self-Check Test #1

Directions — Match the key-word with the missing portion of the statements or definitions. Place the letter of the correct word(s) in the space provided.

A . CONCATENATION	E . MID$( )	I . STRING VARIABLE
B . LEN( )	F . RIGHT$( )	J . VAL( )
C . LEFT$( )	G . STR$( )	
D . MANIPULATION	H . STRING LITERAL	

1. ——— A ( ) is a variable which contains characters.

2. ——— A ( ) is a string of characters within quotes.

3. ——— The ( ) command instructs the computer to count the number of characters in a string.

4. ——— The ( ) command will extract a specific number of characters from the left-most portion of a character string.

5. ——— The ( ) will extract a specific number of characters from any specified location within a character string.

6. ——— Of these three string functions, LEFT$, MID$, and RIGHT$, the most useful is ( ).

7. ——— Any two strings or pieces of strings can be put together using a plus sign (+). This function is called ( ).

8. ——— The ( ) function will convert a string to a number for use in calculations.

9. ——— The ( ) function will convert a number into a character to be used in a string.

10. ——— The ( ) function will instruct the computer to start with a specified character and extract the rest of the string.

Lesson Ten Self-Check Test #2

Directions — Choose the answer that BEST completes the statement. Write the letter of your choice in the space provided.

——— 1. Learning to manipulate strings gives more control over

A . the beginning and ending positions of the variables.
B . storing the strings in multiple dimensional arrays.
C . how we set up the DATA-READ lines in a program.
D . how information is displayed on the screen.

——— 2. The statement LEN(A$) would tell you

A . the length of a word stored in variable A$.
B . that A$ is the variable for the array LEN.
C . LEN is a variable assigned to A$.
D . variable LEN is multiplied by subscript A$.

——— 3. Three commands which let you manipulate character strings by actually taking them apart are:

A . STR$( ), LEN( ), LEFT$( )
B . DIM, LEN( ), VAL( )
C. LEFT$( ), RIGHT$( ), MID$( )
D. RIGHT$( ), DIM, LEN( )

——— 4. One of the actual uses of the commands to manipulate characters is to

A . extract the string to use in another subroutine.
B . extract a portion of a date and print it in a more readable form.
C . STEP 2 so spaces will appear between the characters.
D . extract a portion of the string to use in another program.

——— 5. If given the date in this form: A$ = "07041776" the statement:

PRINT LEFT$(A$,2)

will instruct the computer to PRINT

A . 07
B . 0704
C. 1776
D. 04

——— 6. The function that takes a number and converts the number into a string is the

A . LEN( )
B . MID$( )
C. STR$( )
D. LEFT$( )

——— 7. To extract characters one at a time from a string, use a FOR-NEXT loop and

A . LEFT$( )
B . MID$( )
C. RIGHT$( )
D. STR$( )

——— 8. The function used to convert a string to a numeric value is

A . MID$( )
B . DIM
C. STR$( )
D. VAL( )

——— 9. If given the date: A$ = "07041776" the statement:

PRINT RIGHT$(A$,4)

will instruct the computer to PRINT

A . 0704
B . 0417
C. 1776
D. 7041

———10. If given the date: A$ = "07041776" the statement:

PRINT MID$(A$,4)

will instruct the computer to PRINT

A . 0704
B . 41776
C. 041776
D. 1776

Assignment For Lesson Ten

The assignment for Lesson Ten is to write a "NAME GAME" program in which the operator enters a first name and a last name.

After those entries, have the operator enter a birth date as a string. Verify that the birth date was properly entered as an eight character string of numbers: two numbers for the month, two numbers for the day and four numbers for the year. (The VAL ( ) function may be helpful when initially testing to verify that a number was entered.)

After the operator has entered the data, clear the screen and PRINT the formatted birth date at the top of the screen.

Next, PRINT "IN THE YEAR 2000, YOU WILL BE _____ YEARS OLD." Replace the blank with the operator's age in the year 2000. After calculating the operator's age in the year 2000 , store the "2000 age" in a string variable.

Instruct the computer to print the operator's name backwards.

Then use the string functions two or more times to mix up the letters of the name.

After each misspelling of the name, ask the operator if you, the computer, spelled the name correctly.

Finally, PRINT the name correctly and end the program.

Additional Assignment For Lesson Ten, #1

Write a program which allows the operator to enter a phrase from one to nineteen characters in length. Use the LEN( ) command to check the length of the entry.

When a phrase is entered, use the RIGHT$( ) and LEFT$( ) commands to print the first letter of the phrase at position one on line one, and the last letter of the phrase at position forty on line one.

Then print the first and second letters of the phrase on line two at position one and print the last two characters of the phrase at position thirty-nine.

Continue printing one additional letter of the phrase on each screen line until all the letters of the phrase are printed.

This will form the following pattern:

```
T                                                E
TH                                              SE
THI                                            ASE
THIS                                          RASE
THIS                                         HRASE
THIS I                                      PHRASE
THIS IS                                     PHRASE
THIS IS                                   E PHRASE
THIS IS T                                HE PHRASE
THIS IS TH                              THE PHRASE
THIS IS THE                             THE PHRASE
THIS IS THE                           S THE PHRASE
THIS IS THE P                        IS THE PHRASE
THIS IS THE PH                       IS THE PHRASE
THIS IS THE PHR                    S IS THE PHRASE
THIS IS THE PHRA                  IS IS THE PHRASE
THIS IS THE PHRAS                HIS IS THE PHRASE
THIS IS THE PHRASE              THIS IS THE PHRASE
```

When the entire phrase has been printed, ask the operator if he or she wishes to enter another phrase.

If the operator responds "N", clear the screen and end the program. If the operator responds "Y", return to the beginning of the program and allow another phrase to be entered.

(You will find a FOR-NEXT loop in the printing section to be very helpful.)

Additional Assignment For Lesson Ten, #2

Write a "secret message" program which allows an operator to encode and decode messages. Encoding is the process of translating a normal English sentence into a code. Decoding is the process of translating a coded message into understandable English.

To create your code, you will create a special alphabet.

To encode a message, you substitute a letter from the normal alphabet with the corresponding letter from your special alphabet.

To decode a message, you substitute a letter in your special alphabet with the corresponding letter from the normal alphabet.

Begin by dimensioning two character arrays, each with twenty-six elements (twenty-six elements for the letters A-Z).

In the first array you will store the normal alphabet: the characters A-Z.

The second array will store your special alphabet. One of the easiest methods of creating a special alphabet is to reverse the order of the alphabet: the characters Z-A.

To encode a message, you will create a second version of the message using the characters of your special alphabet instead of the normal alphabet.

First allow the operator to enter a message. Then use a loop and the MID$( ) function to compare each of the characters in the message, one at a time, with the characters of the normal alphabet stored in the elements of the first array. When you find a match, substitute the character in the normal alphabet with the character in the same element of the second array which stores your special alphabet.

For example, the letter "A" will be the first element of the normal alphabet. Let's assume that "Z" is stored in the first element of your special alphabet. The coded message would substitute the letter "Z" for each occurrence of the letter "A".

You will create the coded version of the message by concatenating the characters of your special alphabet, one at a time, to a variable. (Note the concatenation used in the course answer to the lesson assignment reversing the name.)

If the operator enters a character other than A-Z, a space or number for example, use that character as it is.

When the message has been encoded, display both the original message, and the encoded version.

To decode a message, the process will be the same as encoding except the characters in the normal alphabet will be substituted for the characters in your special alphabet.

LESSON ELEVEN: *LO RES GRAPHICS*

LESSON OBJECTIVES

In Lesson Eleven you will be introduced to the *lo resolution color graphics (Lo Res)* capabilities of the computer.

You will learn how to place blocks and lines of color at any of the forty rows and forty columns of the *Lo Res screen.*

You will learn how to mix color pictures and printed text on the same screen through the use of the *text window*, the last four lines of the screen.

Finally, you will learn how to change a color on the Lo Res screen.

The new BASIC commands introduced in this lesson are: **GR, COLOR=, PLOT, TEXT, HLIN, VLIN** and **SCRN( )**

Program Listing For Lesson 11

First Plot

```
10  REM :: PGM FOR PLOT
20  REM  SAVE FIRST PLOT
500  HOME
510  GR
1000  REM :: MAIN PGM MODULE
1010  REM :: DISPLAY TEXT WINDOW
1020  VTAB 21
1030  HTAB 1
1040  INPUT "ENTER COLUMN COORDINATE (-1 TO END): ";X
1050  IF X =  - 1 THEN 9000
1060  IF X < 0 OR X > 39 THEN 1020
1070  VTAB 22
1080  HTAB 22
1090  INPUT "ROW: ";Y
1100  IF Y < 0 OR Y > 39 THEN 1070
1110  VTAB 23
1120  HTAB 22
1130  INPUT "COLOR: ";C
1140  IF C < 0 OR C > 15 THEN 1110
1150  COLOR= C
1160  PLOT X,Y
1170  GOTO 1000
9000  REM :: END-OF-JOB
9010  TEXT
9020  HOME
9030  END
```

Bar Chart

```
10  REM :: BAR CHART PGM
20  REM  SAVE BAR CHART
50  DIM C(6)
100  DATA  7,1,9,13,12,9,10,26,3
```

```
110  FOR I = Ø TO 6
120  READ C(I)
130  NEXT I
140  READ X,Y
400  GR
500  COLOR= 15
510  HLIN Ø,39 AT Ø
520  VLIN Ø,36 AT 39
530  HLIN Ø,39 AT 36
540  VLIN 36,Ø AT Ø
999  REM :: MAIN PGM MODULE
1000  FOR I = Ø TO 6
1010  COLOR= C(I)
1020  FOR J = 1 TO 5
1030  VLIN X,35 AT Y
1040 Y = Y + 1
1050  NEXT J
1060 X = X - 4
1070  NEXT I
1100  HOME
1110  VTAB 21
1120  PRINT "    JAN  FEB  MAR  APR  MAY  JUN  JUL"
1130  PRINT
1140  INPUT "PRESS RETURN TO END ";IN$
1150  TEXT
1160  HOME
1170  END
```

SCRN Example

```
10  REM :: FIRST SCRN PROGRAM
20  REM  SAVE SCRN EXAMPLE
1000  HOME : GR
1010  FOR I = Ø TO 15
1020  COLOR= I
1030  HLIN Ø,39 AT I
1040  NEXT I
1100  HOME : VTAB 21
1110  INPUT "ENTER COLUMN (-1=END): ";X
1120  IF X =  - 1 THEN 9000
1130  IF X < Ø OR X > 39 THEN 1100
1140  INPUT "ENTER ROW..: ";Y
1150  IF Y < Ø OR Y > 39 THEN 1140
1160 C =  SCRN( X,Y)
1200  HOME : VTAB 21
1210  PRINT "THE COLOR AT ROW: ";Y;" COLUMN: ";X
1220  PRINT "IS THE COLOR NUMBER ";C
1230  FOR I = 1 TO 5000: NEXT I
1240  COLOR= 15: PLOT X,Y
1250  GOTO 1100
9000  TEXT : HOME
9010  END
```

Dragon

```
10  REM :: DRAGON PICTURE ::
20  REM  SAVE DRAGON
500  HOME
510  GR
890  REM :: BORDER ::
900  COLOR= 8
910  HLIN 0,39 AT 0
920  HLIN 0,39 AT 38
930  VLIN 0,38 AT 0
940  VLIN 0,38 AT 39
950  COLOR= 13
960  FOR I = 24 TO 37
970  HLIN 1,38 AT I
980  NEXT I
990  REM :: THE DRAGON ::
1000  COLOR= 12
1010  HLIN 13,19 AT 6
1020  HLIN 13,21 AT 7
1030  HLIN 12,23 AT 8
1040  HLIN 11,24 AT 9
1050  HLIN 11,21 AT 10
1060  HLIN 11,19 AT 11
1070  HLIN 10,18 AT 12
1080  HLIN 10,18 AT 13
1090  HLIN 9,19 AT 14
1100  HLIN 9,23 AT 15
1110  HLIN 8,24 AT 16
1120  HLIN 8,25 AT 17
1130  HLIN 7,26 AT 18
1140  HLIN 7,26 AT 19
1150  HLIN 6,25 AT 20
1160  HLIN 5,21 AT 21
1170  HLIN 4,21 AT 22
1180  HLIN 3,21 AT 23
1190  HLIN 2,22 AT 24
1200  HLIN 1,23 AT 25
1210  HLIN 1,24 AT 26
1220  HLIN 1,25 AT 27
1230  HLIN 1,24 AT 28
1240  HLIN 1,16 AT 29
1250  HLIN 1,15 AT 30
1260  HLIN 1,14 AT 31
1270  HLIN 1,14 AT 32
1280  HLIN 1,14 AT 33
1290  HLIN 1,13 AT 34
1300  HLIN 1,11 AT 35
1310  HLIN 1,10 AT 36
1320  HLIN 1,9 AT 37
1990  REM :: THE PAW
2000  VLIN 36,37 AT 15
2010  VLIN 34,37 AT 16
2020  VLIN 33,37 AT 17
2030  VLIN 32,37 AT 18
2040  VLIN 32,37 AT 19
2050  VLIN 32,34 AT 20
2060  VLIN 32,36 AT 21
2070  VLIN 32,34 AT 22
2080  VLIN 32,34 AT 23
2090  VLIN 32,36 AT 24
2100  VLIN 32,33 AT 25
2110  VLIN 32,33 AT 26
2120  VLIN 32,33 AT 27
2130  PLOT 26,22
2990  REM :: HIGHLIGHTS
3000  COLOR= 4
3010  VLIN 20,24 AT 1
3020  VLIN 19,23 AT 2
3030  VLIN 18,22 AT 3
3040  VLIN 17,21 AT 4
3050  VLIN 16,20 AT 5
3060  VLIN 14,19 AT 6
3070  VLIN 12,17 AT 7
3080  VLIN 10,15 AT 8
3090  VLIN 8,13 AT 9
3100  VLIN 7,11 AT 10
3110  VLIN 7,8 AT 11
3120  VLIN 6,7 AT 12
3130  VLIN 1,3 AT 12
3140  VLIN 3,5 AT 13
3150  VLIN 10,12 AT 14
3160  VLIN 12,13 AT 19
3170  VLIN 11,14 AT 20
3180  VLIN 14,15 AT 21
3190  VLIN 10,11 AT 22
3200  VLIN 20,23 AT 22
3210  VLIN 23,24 AT 23
3220  VLIN 20,21 AT 24
3230  VLIN 24,25 AT 24
3240  VLIN 21,22 AT 25
3250  VLIN 25,26 AT 25
3260  VLIN 20,21 AT 26
3270  VLIN 22,23 AT 27
3280  PLOT 13,11
3290  PLOT 14,4
3300  PLOT 15,5
3310  PLOT 20,5
3320  PLOT 22,6
3330  PLOT 24,7
3340  PLOT 25,9
3350  PLOT 21,11
3360  PLOT 22,14
3370  PLOT 23,10
3380  PLOT 24,10
3390  PLOT 27,19
3400  PLOT 26,23
3410  PLOT 26,26
3420  PLOT 28,23
3430  PLOT 21,37
3440  PLOT 24,36
3450  PLOT 27,34
3460  PLOT 23,20
3470  PLOT 14,5
3480  PLOT 14,6
3990  REM :: MORE BACKGROUND
4000  COLOR= 13
```

```
4010  VLIN 25,28 AT 18
4020  VLIN 26,28 AT 19
4030  VLIN 26,28 AT 20
4040  VLIN 27,28 AT 21
4100  COLOR= 0
4110  PLOT 17,6
4120  VLIN 6,8 AT 16
4490  REM :: THE EYE
4500  COLOR= 1
4510  HLIN 21,22 AT 12
4520  COLOR= 15
4530  HLIN 21,22 AT 13
4990  REM :: THE SMOKE
5000  COLOR= 10
5010  VLIN 20,21 AT 27
5020  VLIN 20,22 AT 28
5030  VLIN 19,22 AT 29
5040  VLIN 18,19 AT 30
5050  VLIN 17,22 AT 31
5060  VLIN 16,19 AT 32
5070  VLIN 17,20 AT 33
5080  PLOT 30,21
5990  REM :: THE FIRE
6000  COLOR= 9
6010  VLIN 21,22 AT 23
6020  VLIN 22,23 AT 24
6030  VLIN 23,24 AT 25
6040  VLIN 24,25 AT 26
6050  VLIN 24,25 AT 27
6060  VLIN 24,26 AT 28
6070  VLIN 23,26 AT 29
6080  VLIN 22,23 AT 30
6090  VLIN 25,28 AT 30
6100  VLIN 23,29 AT 31
6110  VLIN 20,23 AT 32
6120  VLIN 25,27 AT 32
6130  VLIN 21,22 AT 33
6140  VLIN 24,25 AT 33
6150  VLIN 18,24 AT 34
6160  VLIN 21,26 AT 35
6170  VLIN 17,20 AT 36
6180  VLIN 22,23 AT 36
6190  VLIN 16,23 AT 37
6200  VLIN 25,26 AT 37
6210  VLIN 19,20 AT 38
6220  VLIN 22,23 AT 38
6230  VLIN 25,26 AT 38
6240  PLOT 35,19
6250  PLOT 36,25
6990  REM :: PICTURE TITLE
7000  VTAB 21
7010  SPEED= 100
7020  PRINT " *****  T H E   D R A G O N  *****"
7030  SPEED= 255
7040  FOR I = 1 TO 740
7050  NEXT I
8000  VTAB 23
8010  INPUT "DO YOU WISH TO CHANGE A COLOR? (Y/N): ";AN$
```

```
8020  IF  LEFT$ (AN$,1) = "N" THEN 8140
8030  IF  LEFT$ (AN$,1) <  > "Y" THEN 8000
8040  HOME
8050  INPUT "ENTER COLOR # TO CHANGE: ";C0
8060  INPUT "ENTER NEW COLOR .......: ";C
8070  COLOR= C
8080  FOR I = 1 TO 37
8090  FOR J = 1 TO 38
8100  IF  SCRN( J,I) = C0 THEN  PLOT J,I
8110  NEXT J
8120  NEXT I
8130  GOTO 8000
8140  TEXT
8150  HOME
8160  END
```

Lesson Eleven Self-Check Test #1

Directions — Match the correct item at the bottom of the page with the statement or definition. Place the letter of the choice in the space provided. The items can be used only once. There are more items than statements.

1. ——— The computer has ( ) modes for displaying colors.
2. ——— LO-RES uses small ( ) of color.
3. ——— HI-RES uses small ( ) of color.
4. ——— For LO-RES graphics there are ( ) colors available.
5. ——— Prior to this lesson, all programming has used characters on the 24 screen lines. Displaying information on the 24 screen lines is called ( ).
6. ——— ( ) instructs the computer to switch to LO-RES graphics.
7. ——— The ( ) command instructs the computer to use a specified color.
8. ——— The ( ) command draws a horizontal line.
9. ——— While in LO-RES graphics, the computer is in ( ).
10. ——— The ( ) command identifies the color that is at a given location on the LO-RES screen.
11. ——— The ( ) command draws a vertical line.
12. ——— After you have worked in the graphics mode you can instruct the computer to return to the regular 24 line screen with the ( ) command.
13. ——— In the graphics mode the four lines at the bottom of the screen are called the ( ).
14. ——— The ( ) command instructs the computer to draw a single block of color at a specified set of coordinates.
15. ——— The graphics screen is similar to ( ).

A . BLOCKS
B . COLOR=
C . COLOR$=
D . DOTS
E . SIXTEEN
F . GR
G . GR$
H . GRAPH PAPER
I . HLIN
J . PERCENTAGE
K . PLOT
L . SCR$( )
M . SCRN( )
N . SCRN$( )
O . GRAPHICS MODE
P . TEXT
Q . TEXT MODE
R . TEXT WINDOW
S . THREE
T . TWO
U . VLIN
V . VLIN$

Lesson Eleven Self-Check Test #2

Directions — Choose the answer that BEST completes the statement. Write the letter of your choice in the space provided.

——— 1. The columns and rows of the LO-RES graphics mode screen are

A . 1-40
B . 0-50
C . 1-25
D . 0-10
E . 0-39

——— 2. In LO-RES graphics a position on the screen is described by

A . two numbers separated by a colon.
B . two numbers separated by a comma.
C . two numbers separated by quotation marks.
D . two numbers separated by a period.
E . two numbers separated by a semicolon.

——— 3. The menu selection "Standard Color Names" shows the color bars on the screen. The names are

A . abbreviated and written vertically above the colors.
B . abbreviated and written horizontally below the colors.
C . abbreviated and written horizontally above the colors.
D . abbreviated and written vertically below the colors.

——— 4. The number of plotting positions on the LO-RES screen is

A . 255
B . 24,000
C . 21 * 24
D . 1600
E . 1040

——— 5. Programs can be written to draw any of the colors at any position on the screen by instructing the computer to draw in a given color at

A . the top and bottom.
B . the column and row.
C . the length and width
D . the graphic screen and the text window.

——— 6. The syntax for the color command is

A . COLOR = X
B . X = COLOR$
C . COLOR$ = X,Y
D . COLOR = X,Y

——— 7. When in the Graphics Mode, messages are written to the operator in the text lines which are located at lines

A . 1-20
B . 19-24
C . 21-24
D . 20-23

——— 8. The command line:

HLIN 0,20 AT 5

instructs the computer to

A . hold the horizontal line from column to column.
B . draw a horizontal line from row zero to row 20 at column 5.
C . draw a horizontal line from column zero to column twenty at row five.
D . draw a horizontal line to column five from column zero and column twenty.

——— 9. If you fail to return the computer to TEXT MODE at the end of a program, the color will remain on the screen and

A . the computer will give you an error message to change diskettes.
B . the program will not be useable again.
C . the text will be blocked out and not readable.
D . your entire text will scroll through the Text Window.
E . you will have to reboot the Master System Diskette.

——— 10. To use HLIN and VLIN you must be sure to

A . have your coordinates in order.
B . use the COLOR command before you use them.
C . have the Text Mode activated.
D . assign the coordinates to arrays in order.

——— 11. The command: PLOT 5,12 will place a block of color at

A . row 5, column 12
B . row 5 to row 12
C . column 5, row 12
D . column 5 to column 12

Assignment For Lesson Eleven

You are to write a program which draws a Lo Res picture.

First draw the picture on a Lo Res screen grid. Then identify the coordinates for the rows and columns and the color numbers for the PLOT, HLIN, VLIN and COLOR= commands.

When drawing the picture, your program must include HLIN, VLIN and PLOT commands.

After the picture is drawn, allow the operator to change a color or end the program.

If the operator wishes to change a color, prompt the operator to enter the number of the color he or she wishes to change and the number of the new color. Then use the SCRN( ) command to change all occurrences of the current color to the new color specified by the operator.

Additional Assignment For Lesson Eleven, #1

Write a program which draws a diamond of color 9 rows tall and 9 columns wide in the center of the screen.

Make the two outside rows of the diamond a different color than the interior to create a border.

When the first diamond has been drawn on the screen, allow the operator to change both of the colors or end the program.

Unlike the color change example in the lesson which replaces only one color, you will need to store two original colors and two new colors and use the SCRN( ) command twice.

You may use a color change routine which examines a smaller portion of the screen than the routine used in the lesson to make the color change operations faster.

Additional Assignment For Lesson Eleven, #2

Write a program which draws rectangles of color from the middle of the screen to the outside of the screen.

Start with a small rectangle in the middle of the screen and draw increasingly larger rectangles until you reach the outside of the screen.

Use any pattern of colors you wish for the graphic display.

LESSON TWELVE: *ANIMATION*

LESSON OBJECTIVES

In Lesson Twelve you will be introduced to the concepts of computer animation.

Computer animation creates the *illusion of movement* by repeatedly drawing, and erasing a block of color at one location, and redrawing the block at another location.

An animated block of color is *"erased"* by redrawing in the background color.

You will learn the use of the computer's *random number function* and the use of the *integer function* to convert a decimal number to a whole number.

You will learn how to instruct the computer to read the contents of any memory location, and how to instruct the computer to put information directly into a memory location.

You will learn another way to allow the operator to enter information from the keyboard and how to make sounds other than the bell sound.

The new BASIC commands introduced in this lesson are: **RND( ), INT( ), PEEK( ),** and **POKE.**

Program Listing For Lesson 12

First Animation: Version #1

```
10  REM :: FIRST ANIMATION PGM
20  REM  SAVE FIRST ANIMATION
100 C = 15
200 B = 0
1000  HOME : GR
1100  FOR I = 1 TO 39
1200  COLOR= C
1300  PLOT I,20
1400  COLOR= B
1500  PLOT I,20
1600  NEXT I
2000  VTAB 21
2010  INPUT "ENTER E TO END ";A$
2020  IF A$ <  > "E" THEN 1000
9000  TEXT : HOME
9010  END
```

First Animation: Version #2

```
10  REM :: FIRST ANIMATION PGM
20  REM  SAVE FIRST ANIMATION
100 C = 15
200 B = 0
1000  HOME : GR
1100  FOR I = 1 TO 39
```

```
12ØØ  COLOR= C
13ØØ  PLOT I,2Ø
14ØØ  COLOR= B
15ØØ  PLOT I - 1,2Ø
16ØØ  NEXT I
2ØØØ  VTAB 21
2Ø1Ø  INPUT "ENTER E TO END ";A$
2Ø2Ø  IF A$ <  > "E" THEN 1ØØØ
9ØØØ  TEXT : HOME
9Ø1Ø  END
```

First Animation: Version #3

```
1Ø  REM :: FIRST ANIMATION PGM
2Ø  REM  SAVE FIRST ANIMATION
1ØØ C = 15
2ØØ B = Ø
1ØØØ  HOME : GR
11ØØ  FOR I = 1 TO 39
12ØØ  COLOR= C
13ØØ  PLOT I,2Ø
14ØØ  COLOR= B
15ØØ  PLOT I - 1,2Ø
16ØØ  NEXT I
17ØØ  PLOT 39,2Ø
2ØØØ  VTAB 21
2Ø1Ø  INPUT "ENTER E TO END ";A$
2Ø2Ø  IF A$ <  > "E" THEN 1ØØØ
9ØØØ  TEXT : HOME
9Ø1Ø  END
```

First Animation: Final Version

```
1Ø  REM :: FIRST ANIMATION PGM
2Ø  REM  SAVE FIRST ANIMATION
1ØØ C = 15
2ØØ B = Ø
1ØØØ  HOME : GR
11ØØ  FOR I = 1 TO 39
12ØØ  COLOR= C
13ØØ  PLOT I,2Ø
135Ø  GOSUB 91ØØ
14ØØ  COLOR= B
15ØØ  PLOT I - 1,2Ø
155Ø  GOSUB 91ØØ
16ØØ  NEXT I
17ØØ  PLOT 39,2Ø
2ØØØ  VTAB 21
2Ø1Ø  INPUT "ENTER E TO END ";A$
2Ø2Ø  IF A$ <  > "E" THEN 1ØØØ
9ØØØ  TEXT : HOME
9Ø1Ø  END
91ØØ  FOR J = 1 TO 1ØØ: NEXT J: RETURN
```

Random Numbers: Version #1

```
10  FOR I = 1 TO 10
20  PRINT  RND (1)
30  NEXT I
```

Random Numbers: Version #2

```
10  FOR I = 1 TO 10
20  PRINT 4 *  RND (1) + 1
30  NEXT I
```

Random Numbers: Version #3

```
10  FOR I = 1 TO 10
20  PRINT 20 *  RND (1) + 1
30  NEXT I
```

Random Numbers: Final Version

```
10  FOR I = 1 TO 15
20  PRINT  INT (4 *  RND (1) + 1)
30  NEXT I
```

Roaming Block

```
10  REM :: RANDOM MOVEMENT PGM
20  REM  SAVE ROAMING BLOCK
40  REM :: HOUSEKEEPING SECTION
100 C = 11
110 B = 0
490  REM :: STARTING POSITION
500 X = 20:Y = 20
510 XP = X:YP = Y
600  HOME : GR
1000  FOR I = 1 TO 200
1010 D =  INT (4 *  RND (1) + 1)
1080  ON D GOTO 1100,1200,1300,1400
1090  REM :: DIRECTION 1=UP
1100 Y = Y - 1: IF Y < 0 THEN Y = 39
1110  GOTO 1500
1190  REM :: DIRECTION 2=LEFT
1200 X = X - 1: IF X < 0 THEN X = 39
1210  GOTO 1500
1290  REM :: DIRECTION 3=DOWN
1300 Y = Y + 1: IF Y > 39 THEN Y = 0
1310  GOTO 1500
1390  REM :: DIRECTION 4=RIGHT
1400 X = X + 1: IF X > 39 THEN X = 0
1500  COLOR= C: PLOT X,Y
1510  COLOR= B: PLOT XP,YP
1520 XP = X:YP = Y
```

```
2000  NEXT I
9000  VTAB 21: INPUT "PRESS RETURN ";A$
9010  TEXT : HOME
9020  END
```

Keyboard Peek

```
10  REM :: KEYBOARD PEEK PGM
20  REM  SAVE KEYBOARD PEEK
1000 PX =  PEEK ( - 16384)
1010  PRINT PX
1020  GOTO 1000
```

Roaming Block Peek

```
10  REM :: RANDOM MOVEMENT PGM
20  REM  SAVE ROAMING BLOCK PEEK
40  REM :: HOUSEKEEPING SECTION
100 C = 11
110 B = 0
120 AR = 149:AL = 136:SB = 160
130 KB =  - 16384:KS =  - 16368
490  REM :: STARTING POSITION
500 X = 20:Y = 20
510 XP = X:YP = Y
600  HOME : GR
1000  REM :: BEGIN ANIMATION
1010 D =  INT (4 *  RND (1) + 1)
1020 PK =  PEEK (KB)
1030  IF PK = AR THEN D = 4: POKE KS,0
1040  IF PK = AL THEN D = 2: POKE KS,0
1050  IF PK = SB THEN 9000
1080  ON D GOTO 1100,1200,1300,1400
1090  REM :: DIRECTION 1=UP
1100 Y = Y - 1: IF Y < 0 THEN Y = 39
1110  GOTO 1500
1190  REM :: DIRECTION 2=LEFT
1200 X = X - 1: IF X < 0 THEN X = 39
1210  GOTO 1500
1290  REM :: DIRECTION 3=DOWN
1300 Y = Y + 1: IF Y > 39 THEN Y = 0
1310  GOTO 1500
1390  REM :: DIRECTION 4=RIGHT
1400 X = X + 1: IF X > 39 THEN X = 0
1500  COLOR= C: PLOT X,Y
1510  COLOR= B: PLOT XP,YP
1520 XP = X:YP = Y
2000  GOTO 1010
9000  VTAB 21: INPUT "PRESS RETURN ";A$
9010  TEXT : HOME
9020  END
```

First Sound

```
10  REM :: SOUND PEEK PROGRAM
20  REM  SAVE FIRST-SOUND
1000  VTAB 20
1010  INPUT "HOW MANY PEEKS? (0=END) ";SN
1020  IF SN = 0 THEN 9000
1030  FOR I = 1 TO SN
1040 X =  PEEK ( - 16336)
1050  NEXT I
1060  GOTO 1000
9000  END
```

Dragonfly

```
10  REM :: DRAGON PICTURE ::
20  REM  SAVE DRAGONFLY
500  HOME
510  GR
890  REM :: BORDER ::
900  COLOR= 8
910  HLIN 0,39 AT 0
920  HLIN 0,39 AT 38
930  VLIN 0,38 AT 0
940  VLIN 0,38 AT 39
950  COLOR= 13
960  FOR I = 24 TO 37
970  HLIN 1,38 AT I
980  NEXT I
990  REM :: THE DRAGON ::
1000  COLOR= 12
1010  HLIN 13,19 AT 6
1020  HLIN 13,21 AT 7
1030  HLIN 12,23 AT 8
1040  HLIN 11,24 AT 9
1050  HLIN 11,21 AT 10
1060  HLIN 11,19 AT 11
1070  HLIN 10,18 AT 12
1080  HLIN 10,18 AT 13
1090  HLIN 9,19 AT 14
1100  HLIN 9,23 AT 15
1110  HLIN 8,24 AT 16
1120  HLIN 8,25 AT 17
1130  HLIN 7,26 AT 18
1140  HLIN 7,26 AT 19
1150  HLIN 6,25 AT 20
1160  HLIN 5,21 AT 21
1170  HLIN 4,21 AT 22
1180  HLIN 3,21 AT 23
1190  HLIN 2,22 AT 24
1200  HLIN 1,23 AT 25
1210  HLIN 1,24 AT 26
1220  HLIN 1,25 AT 27
1230  HLIN 1,24 AT 28
1240  HLIN 1,16 AT 29
1250  HLIN 1,15 AT 30
1260  HLIN 1,14 AT 31
1270  HLIN 1,14 AT 32
```

```
1280  HLIN 1,14 AT 33
1290  HLIN 1,13 AT 34
1300  HLIN 1,11 AT 35
1310  HLIN 1,10 AT 36
1320  HLIN 1,9 AT 37
1990  REM :: THE PAW
2000  VLIN 36,37 AT 15
2010  VLIN 34,37 AT 16
2020  VLIN 33,37 AT 17
2030  VLIN 32,37 AT 18
2040  VLIN 32,37 AT 19
2050  VLIN 32,34 AT 20
2060  VLIN 32,36 AT 21
2070  VLIN 32,34 AT 22
2080  VLIN 32,34 AT 23
2090  VLIN 32,36 AT 24
2100  VLIN 32,33 AT 25
2110  VLIN 32,33 AT 26
2120  VLIN 32,33 AT 27
2130  PLOT 26,22
2990  REM :: HIGHLIGHTS
3000  COLOR= 4
3010  VLIN 20,24 AT 1
3020  VLIN 19,23 AT 2
3030  VLIN 18,22 AT 3
3040  VLIN 17,21 AT 4
3050  VLIN 16,20 AT 5
3060  VLIN 14,19 AT 6
3070  VLIN 12,17 AT 7
3080  VLIN 10,15 AT 8
3090  VLIN 8,13 AT 9
3100  VLIN 7,11 AT 10
3110  VLIN 7,8 AT 11
3120  VLIN 6,7 AT 12
3130  VLIN 1,3 AT 12
3140  VLIN 3,5 AT 13
3150  VLIN 10,12 AT 14
3160  VLIN 12,13 AT 19
3170  VLIN 11,14 AT 20
3180  VLIN 14,15 AT 21
3190  VLIN 10,11 AT 22
3200  VLIN 20,23 AT 22
3210  VLIN 23,24 AT 23
3220  VLIN 20,21 AT 24
3230  VLIN 24,25 AT 24
3240  VLIN 21,22 AT 25
3250  VLIN 25,26 AT 25
3260  VLIN 20,21 AT 26
3270  VLIN 22,23 AT 27
3280  PLOT 13,11
3290  PLOT 14,4
3300  PLOT 15,5
3310  PLOT 20,5
3320  PLOT 22,6
3330  PLOT 24,7
3340  PLOT 25,9
3350  PLOT 21,11
3360  PLOT 22,14
3370  PLOT 23,10
3380  PLOT 24,10
3390  PLOT 27,19
3400  PLOT 26,23
3410  PLOT 26,26
3420  PLOT 28,23
3430  PLOT 21,37
3440  PLOT 24,36
3450  PLOT 27,34
3460  PLOT 23,20
3470  PLOT 14,5
3480  PLOT 14,6
3990  REM :: MORE BACKGROUND
4000  COLOR= 13
4010  VLIN 25,28 AT 18
4020  VLIN 26,28 AT 19
4030  VLIN 26,28 AT 20
4040  VLIN 27,28 AT 21
4100  COLOR= 0
4110  PLOT 17,6
4120  VLIN 6,8 AT 16
4490  REM :: THE EYE
4500  COLOR= 1
4510  HLIN 21,22 AT 12
4520  COLOR= 15
4530  HLIN 21,22 AT 13
4990  REM :: THE SMOKE
5000  COLOR= 10
5010  VLIN 20,21 AT 27
5020  VLIN 20,22 AT 28
5030  VLIN 19,22 AT 29
5040  VLIN 18,19 AT 30
5050  VLIN 17,22 AT 31
5060  VLIN 16,19 AT 32
5070  VLIN 17,20 AT 33
5080  PLOT 30,21
5990  REM :: THE FIRE
6000  COLOR= 9
6010  VLIN 21,22 AT 23
6020  VLIN 22,23 AT 24
6030  VLIN 23,24 AT 25
6040  VLIN 24,25 AT 26
6050  VLIN 24,25 AT 27
6060  VLIN 24,26 AT 28
6070  VLIN 23,26 AT 29
6080  VLIN 22,23 AT 30
6090  VLIN 25,28 AT 30
6100  VLIN 23,29 AT 31
6110  VLIN 20,23 AT 32
6120  VLIN 25,27 AT 32
6130  VLIN 21,22 AT 33
6140  VLIN 24,25 AT 33
6150  VLIN 18,24 AT 34
6160  VLIN 21,26 AT 35
6170  VLIN 17,20 AT 36
6180  VLIN 22,23 AT 36
6190  VLIN 16,23 AT 37
6200  VLIN 25,26 AT 37
6210  VLIN 19,20 AT 38
6220  VLIN 22,23 AT 38
```

```
6230  VLIN 25,26 AT 38
6240  PLOT 35,19
6250  PLOT 36,25
6990  REM :: PICTURE TITLE
7000  VTAB 21
7010  SPEED= 100
7020  PRINT " *****  T H E   D R A G O N  *****"
7030  SPEED= 255
7990  REM :: ADD THE FLY
8000 C = 15
8010 AR = 149:AL = 136:SB = 160
8020 KB =  - 16384:KS =  - 16368
8030 SD =  - 16336
8040  REM :: STARTING POSITION
8050 X = 20:Y = 20
8060 XP = X:YP = Y
8070 B =  SCRN( 20,20)
8080  REM :: BEGIN ANIMATION
8090 D =  INT (4 *  RND (1) + 1)
8100 PK =  PEEK (KB)
8110  IF PK = AR THEN D = 4: POKE KS,0
8120  IF PK = AL THEN D = 2: POKE KS,0
8130  IF PK = SB THEN 9000
8140 S =  PEEK (SD):S =  PEEK (SD)
8150  ON D GOTO 8170,8200,8230,8260
8160  REM :: DIRECTION 1=UP
8170 Y = Y - 1: IF Y < 0 THEN Y = 39
8180  GOTO 8270
8190  REM :: DIRECTION 2=LEFT
8200 X = X - 1: IF X < 0 THEN X = 39
8210  GOTO 8270
8220  REM :: DIRECTION 3=DOWN
8230 Y = Y + 1: IF Y > 39 THEN Y = 0
8240  GOTO 8270
8250  REM :: DIRECTION 4=RIGHT
8260 X = X + 1: IF X > 39 THEN X = 0
8270 TB =  SCRN( X,Y)
8280  COLOR= C: PLOT X,Y
8290  COLOR= B: PLOT XP,YP
8300 XP = X:YP = Y
8310 B = TB
8320 S =  PEEK (SD):S =  PEEK (SD)
8330  GOTO 8090
9000  VTAB 23: INPUT "PRESS RETURN ";A$
9010  TEXT : HOME
9020  END
```

Lesson Twelve Self-Check Test #1

Directions — Choose the word(s) at the bottom of the page that BEST completes the statement. Write the letter of your choice in the space provided. Some of the words may be used more than once or not at all.

——— 1. When creating the illusion of animation on the computer you draw, erase, and redraw. You erase by redrawing in the ( ).

——— 2. The ( ) can be used to slow down the animation process.

——— 3. If you want an object to appear to move in front of a background of several colors, you must first use the ( ) commands to determine the color of the background.

——— 4. The random command is used to have the computer select random numbers. The syntax of the random command is ( ).

——— 5. To select a random row or column for a LO-RES screen a number between ( ) and ( ) is needed.

——— 6. The ( ) command only slows the printing of words on the text screen. It has no effect on the graphics screen.

——— 7. The special function which removes the decimal portion of a number leaving only the integer is ( ).

——— 8. To select one of four random directions for computer animation, you would use the calculation ( ).

——— 9. The memory locations in a computer are called ( ).

——— 10. The processor reads the contents of a given location with the ( ) command.

——— 11. Each key has its own unique number. The computer uses the number to determine which ( ) to display.

——— 12. A number can be placed into a given location by using the ( ) command.

——— 13. The ( ) command is used to modify the bell sound and create a buzzing sound.

——— 14. The ( ) can be used to accept information from the operator without stopping the flow of the program.

A . ADDRESSES
B . CHARACTER
C . BACKGROUND COLOR
D . DECIMAL NUMBER
E . INT( )
F . 1-40
G . PEEK( )
H . POKE
I . RND( )
J . SCRN( )
K . SPEED=
L . TIMING LOOP
M . D=INT(4*RND(1)+1)
N . 0-39
O . D=INT(1*RND(1)+4)

Lesson Twelve Self-Check Test #2

Directions — Choose the answer that BEST completes the statement. Write the letter of your choice in the space provided.

——— 1. As you draw, erase and redraw to create motion of an object on the CRT, it is important to remember

A. the background should be erased with the color black.
B. the object should be begun in columns zero and thirty-nine.
C. the erasing color will be the same color as the background.
D. the foreground color should be the same color as the background.

——— 2. To create the illusion of fluid motion,

A. first PLOT the color at its new coordinates then erase the color at the old coordinates.
B. first erase the block of color, then PLOT the color at its new coordinates.
C. use the HLIN command instead of the PLOT command.
D. always use black as the erasing color.

——— 3. To create the illusion of movement, first

A. erase color where the object has been.
B. PLOT color where the object has been.
C. PLOT color where the object is going to be.
D. store the background color where the object is going to be.

——— 4. In the RND( ) command, the number inside the parentheses

A. cannot be a number larger than 39.
B. can be omitted if random number is over 9.
C. should be between one and nine.
D. has to be a number between zero and fifteen.

——— 5. In this programming line:

```
20 PRINT 4 * RND (1) + 1
```

the range of the random number is determined by

A. the line number "20".
B. the multiplier "4".
C. the multiplier "1".
D. the addend "1".

——— 6. In a program in which the computer selects the direction of movement, the range of whole numbers needed will be calculated by combining

A. the PEEK command and the POKE command.
B. the INT function with the RND function.
C. the RND function and the SCRN command.
D. the INT function and the INPUT command.

——— 7. In animation you want the program to respond to the operator but

A. you don't want the program to wait for the operator.
B. only if the operator has full control of the program.
C. you don't want the operator to lose control.
D. only if the program has full control.

——— 8. The POKE -16368,0 command

A. stops the flow of the program so you can decide what to do.
B. resets the keyboard so you can PEEK for another character.
C. moves an object on the screen by receiving moves from the operator.
D. resets the CRT and gives the operator the address of the move.

——— 9. To move a block of color up the screen,

A. add one to the column coordinate.
B. subtract one from the column coordinate.
C. add one to the row coordinate.
D. subtract one from the row coordinate.

——— 10. To move a block of color to the right,

A. add one to the column coordinate.
B. subtract one from the column coordinate.
C. add one to the row coordinate.
D. subtract one from the row coordinate.

Lesson Twelve Self-Check Test #3

Directions — The following statements are a series of steps to be followed to move a color (white) over a multicolored background color. Put them in the correct order by writing the letter of your choice in the numbered spaces provided.

You begin the animation step with the previous coordinates stored in the variables XP,YP and the erasing background color in the variable B. The current coordinates have been selected and stored in the variables X,Y.

1. ——— A. Use the command: COLOR = B to set the erasing color to the correct background color. Erase by plotting at the previous coordinates: PLOT XP,YP

2. ——— B. Use the command: TB = SCRN(X,Y) to store the background color at the current coordinates.

3. ——— C. Plan ahead by storing the current coordinates and background color for erasing during the next animation step using: XP = X YP = Y B = TB

4. ——— D. Use the command: COLOR = 15 to set the PLOTting color. PLOT the white block at the current coordinates: PLOT X,Y

Assignment For Lesson Twelve

The assignment for this lesson is to add animation and sound to a Lo Res picture.

You may use a picture you have already drawn or create a new picture.

After the program has drawn the picture, use the animation techniques to move a block of color over the background. Be sure and use the SCRN( ) command to store the background color so you can "erase" the animated block.

Use the keyboard PEEK( ) command, "PEEK (-16384)", to allow the operator to enter information from the keyboard. Change the color or direction of the animated block based on the keys pressed by the operator. Remember to use the keyboard POKE command, "POKE -16368,0", to reset the keyboard after the program has found an appropriate key.

Use the sound PEEK command, "PEEK (-16336)", to make the computer buzz as the block of color moves.

You must use the RND( ) and INT( ) functions at least once in the program.

Additional Assignment For Lesson Twelve, #1

Write a program which moves a block of color from the top of the screen to the bottom, or from the bottom of the screen to the top.

When the block reaches the bottom or top of the screen, make the block "wrap around" the screen. If the block moves off the bottom, make it reappear at the top. If the block moves off the top of the screen, make it reappear at the bottom.

Use the keyboard PEEK( ) command and the arrow-right and arrow-left keys to change the direction of the blocks of color. (Arrow-right for down, and arrow-left for up.) If the operator presses the space bar, end the program.

Be sure to reset the keyboard with the keyboard POKE command after finding one of the keys for which you are testing.

HINT: You will need to subtract from the row coordinate to make the block move up and add to the row coordinate to make the block move down.

Additional Assignment For Lesson Twelve, #2

Write a game program in which the operator moves a white block of color through a field of colored blocks from the bottom to the top of the Lo Res screen.

Allow the operator to move the white block by pressing the "I", "J", "K", and "M" keys and to press the space bar to give up.

Do not allow the operator to move the white block off the bottom of the screen.

The operator "wins" when he or she attempts to move the white block off the top of the screen. You will know if the operator has attempted to move the block off the top if the row coordinate is less than zero.

In the housekeeping section, set up the initial field by plotting 40 random blocks of color. (Use colors 1-14. Zero will be the erasing color and 15 the color of the operator's block.)

Use the sound PEEK( ) command to add sound effects to the program.

On each animation step add more blocks of color to the screen making the game more and more difficult.

In this program you will use the SCRN( ) command in a different way. The SCRN( ) command will be used to tell if the operator has moved the white block into a set of coordinates already occupied by another colored block. The erasing color will be black, COLOR = 0.

HINT: Review the course's answer to the animation assignment. You will also find the directional routines in the "ROAMING BLOCK" program to be useful. Modify the "UP" and "DOWN" directions to prevent the operator from moving off the bottom of the screen, and to tell if the operator has successfully moved the white block off the top of the screen.

LESSON THIRTEEN: *DATA FILES*

LESSON OBJECTIVES

In Lesson Thirteen you will be introduced to *sequential* and *random access* data files, also called *"Text Files"*.

You will first learn how to create and use sequential data files to store and retrieve data from diskette. Data in sequential files must be stored and retrieved in order.

You will then learn how to create and use random access files to store and retrieve data. Random access files are comprised of *records*. Data in random access file records may be stored and retrieved in any order.

You will also learn how to prevent a program from stopping when an error occurs by setting an *error trap*.

Learning to work with data files is similar to learning a new dialect of the same language. Some of the command words are similar, but their use and meaning is different.

The special DOS data file commands are proceeded by the command **PRINT CHR$(4)**, followed by the DOS command enclosed in quotation marks.

The new DOS commands introduced in this lesson are: **OPEN, CLOSE, WRITE, PRINT** as a DOS command, **READ** as a DOS command, **INPUT** as a DOS command, **CLOSE, APPEND,** and **ONERR GOTO.**

Program Listing For Lesson 13

Seq. Write

```
10  REM : FIRST SEQENTIAL WRITE
20  REM  SAVE SEQ WRITE
100  DIM N$(2),A(2)
1000  FOR I = 1 TO 2
1010  HOME
1020  INPUT "ENTER NAME ";N$(I)
1030  INPUT "ENTER AGE  ";A(I)
1040  NEXT I
2000  PRINT  CHR$ (4);"OPEN AGES"
2010  PRINT  CHR$ (4);"WRITE AGES"
2020  FOR I = 1 TO 2
2030  PRINT N$(I)
2040  PRINT A(I)
2050  NEXT I
2060  PRINT  CHR$ (4);"CLOSE AGES"
9000  HOME
9010  END
```

Seq. Read

```
10  REM : FIRST SEQUENTIAL READ
20  REM  SAVE SEQ READ
100  DIM N$(2),A(2)
1000  PRINT  CHR$ (4);"OPEN AGES"
1010  PRINT  CHR$ (4);"READ AGES"
```

```
1020  FOR I = 1 TO 2
1030  INPUT N$(I)
1040  INPUT A(I)
1050  NEXT I
1060  PRINT  CHR$ (4);"CLOSE AGES"
1070  HOME
1080  FOR I = 1 TO 2
1090  PRINT "NAME ";N$(I)
1100  PRINT "AGE  ";A(I)
1110  PRINT
1120  NEXT I
1130  INPUT "PRESS RETURN TO END";A$
1140  HOME : END
```

Seq. Append

```
10  REM : FIRST SEQENTIAL APPEND
20  REM  SAVE SEQ APPEND
100  DIM N$(2),A(2)
1000  FOR I = 1 TO 2
1010  HOME
1020  INPUT "ENTER NAME ";N$(I)
1030  INPUT "ENTER AGE  ";A(I)
1040  NEXT I
2000  PRINT  CHR$ (4);"APPEND AGES"
2010  PRINT  CHR$ (4);"WRITE AGES"
2020  FOR I = 1 TO 2
2030  PRINT N$(I)
2040  PRINT A(I)
2050  NEXT I
2060  PRINT  CHR$ (4);"CLOSE AGES"
9000  HOME
9010  END
```

Seq. Read After Append

```
10  REM : READ AFTER APPEND
20  REM  SAVE SEQ READ AFTER APPEND
100  DIM N$(4),A(4)
1000  PRINT  CHR$ (4);"OPEN AGES"
1010  PRINT  CHR$ (4);"READ AGES"
1020  FOR I = 1 TO 4
1030  INPUT N$(I)
1040  INPUT A(I)
1050  NEXT I
1060  PRINT  CHR$ (4);"CLOSE AGES"
1070  HOME
1080  FOR I = 1 TO 4
1090  PRINT "NAME ";N$(I)
1100  PRINT "AGE  ";A(I)
1110  PRINT
1120  NEXT I
1130  INPUT "PRESS RETURN TO END";A$
1140  HOME : END
```

Telephone Write

```
10  REM : WRITE DATA TO TELS
20  REM  SAVE TELEPHONE WRITE
1000  HOME
1010  INPUT "ENTER RECORD NUMBER (0=END) ";N
1020  IF N = 0 THEN 9000
1030  INPUT "ENTER NAME ";N$
1040  INPUT "ENTER PHONE # ";T$
2000  PRINT  CHR$ (4);"OPEN TELS,L45"
2010  PRINT  CHR$ (4);"WRITE TELS,R";N
2020  PRINT N$
2030  PRINT T$
2040  PRINT  CHR$ (4);"CLOSE TELS"
2050  GOTO 1000
9000  HOME
9010  END
```

Telephone Read: Version #1

```
10  REM  : READ DATA FROM TELS
20  REM  SAVE TELEPHONE READ
1000  HOME
1010  INPUT "ENTER RECORD NUMBER TO READ (0=END) ";N
1020  IF N = 0 THEN 9000
1030  PRINT  CHR$ (4);"OPEN TELS,L45"
1040  PRINT  CHR$ (4);"READ TELS,R";N
1050  INPUT N$
1060  INPUT T$
1070  PRINT  CHR$ (4);"CLOSE TELS"
2000  VTAB 5
2010  PRINT "NAME ";N$
2020  PRINT "PHONE ";T$
2030  VTAB 10
2040  INPUT "PRESS RETURN";A$
2050  GOTO 1000
9000  HOME
9010  END
```

Telephone Read: Final Version

```
10  REM  : READ DATA FROM TELS
20  REM  SAVE TELEPHONE READ
900  ONERR  GOTO 7000
1000  HOME
1010  INPUT "ENTER RECORD NUMBER TO READ (0=END) ";N
1020  IF N = 0 THEN 9000
1030  PRINT  CHR$ (4);"OPEN TELS,L45"
1040  PRINT  CHR$ (4);"READ TELS,R";N
1050  INPUT N$
1060  INPUT T$
1070  PRINT  CHR$ (4);"CLOSE TELS"
2000  VTAB 5
2010  PRINT "NAME ";N$
2020  PRINT "PHONE ";T$
2030  VTAB 10
```

```
2040  INPUT "PRESS RETURN";A$
2050  GOTO 1000
7000  PRINT  CHR$ (4);"CLOSE TELS"
7010  VTAB 23
7020  PRINT "RECORD DOES NOT EXIST"
7030  FOR I = 1 TO 2 * 740
7040  NEXT I
7050  GOTO 1000
9000  HOME
9010  END
```

Phone Write

```
10  REM : WRITE DATA TO TELS
20  REM  SAVE PHONE WRITE
300  ONERR  GOTO 500
400  REM  :: GET NUMBER OF RECS
410  PRINT  CHR$ (4);"OPEN TELS,L45"
420  PRINT  CHR$ (4);"READ TELS,R0"
430  INPUT N
440  GOTO 900
490  REM :: OR INITIALIZE FILE
500  PRINT  CHR$ (4);"WRITE TELS,R0"
510  PRINT 0
900  PRINT  CHR$ (4);"CLOSE TELS"
1000  HOME
1010  PRINT "ENTER DATA FOR RECORD #";N + 1
1020  INPUT "ENTER NAME OR 'E' TO END: ";N$
1030  IF N$ = "E" THEN 9000
1040  INPUT "ENTER PHONE # ";T$
1050 N = N + 1
2000  PRINT  CHR$ (4);"OPEN TELS,L45"
2010  PRINT  CHR$ (4);"WRITE TELS,R";N
2020  PRINT N$
2030  PRINT T$
2040  PRINT  CHR$ (4);"CLOSE TELS"
2050  GOTO 1000
9000  PRINT  CHR$ (4);"OPEN TELS,L45"
9010  PRINT  CHR$ (4);"WRITE TELS,R0"
9020  PRINT N
9030  PRINT  CHR$ (4);"CLOSE TELS"
9040  HOME : END
```

Lesson Thirteen Self-Check Test #1

Directions — If the statement is TRUE, place a "T" in the space provided. If the answer is FALSE, place an "F" in the space and change the italicized portion to make the statement TRUE.

——— 1. Programs are ***temporarily*** stored in program files on diskettes.

——— 2. Large quantities of information can be stored for long periods of time in ***data files.***

——— 3. Each DOS command placed within a program line should have this structure:
1000 ***PRINT CHR$(4);"command"***

——— 4. A CATALOG listing of data files displays an ***"S"*** in front of file size.

——— 5. With a sequential file, information must be retrieved in the ***same*** order in which it was stored.

——— 6. ***Random access*** files are the more flexible of the two types of files.

——— 7. You can work with records in a random file in ***any order*** you wish.

——— 8. With a random file you can ***update*** information in a specific record without erasing or changing information in other records.

——— 9. The READ and WRITE commands for a random file ***sometimes*** designate which record you want to use.

Lesson Thirteen Self-Check Test #2

Directions — Locate the word(s) at the bottom of the page that completes the statements. Place the letter of your choice in the space provided. Some of the choices may be used more than once or not at all.

——— 1. Most computers use two types of files. They are ( ).

——— 2. ( ) files only store information which can be retrieved and used in a program.

——— 3. Each character entered is connected to the next character in a ( ) file.

——— 4. Before using a file you must ( ) the file.

——— 5. After a file has been OPENed and data READ from or written to the file, the program should ( ) the file.

——— 6. The ( ) command followed by a PRINT statement is used to store DATA or information in the file.

——— 7. The READ command followed by the ( ) statement will take information from the file for your use.

——— 8. When the computer is instructed to open a ( ) file, the file pointer is positioned at the beginning of the file to add new information.

——— 9. To add new information to an existing sequential file, the ( ) command is used.

——— 10. In a random file each set of information is called a ( ).

A . APPEND
B . CLOSE
C . DATA
D . INPUT
E . OPEN
F . PROGRAM & DATA
G . RANDOM ACCESS
H . RECORD
I . SEQUENTIAL
J . SEQUENTIAL & RANDOM ACCESS
K . WRITE

Lesson Thirteen Self-Check Test #3

Directions — Choose the answer that BEST completes the statement. Write the letter of your choice in the space provided.

——— 1. When you OPEN a RANDOM ACCESS file you tell the computer

A. to position itself at the beginning of the file and add new information.
B. how long you want each record to be.
C. that you will be appending the records.
D. what kind of DATA you want to WRITE and READ.

——— 2. This program line:

1000 PRINT CHR$(4);"OPEN AGES"

is instructing the computer to

A. OPEN the program named AGES.
B. PRINT the program named AGES.
C. PRINT the file named AGES.
D. OPEN the file named AGES.

——— 3. With the program line:

1000 PRINT CHR$(4);"WRITE AGES"

A. all print lines following the command will send information to the file until the file is closed or another command is used.
B. the PRINT statement allows the information to be seen on the CRT for viewing the file named AGES.
C. the file AGES is assigned an array with four elements that will be included in the READ and WRITE statements.
D. the character variable, WRITE AGES is written in a loop 4 times.

——— 4. The letter "L" stands for record length and is followed by the maximum number of characters each record in the file will hold. It is

A. instructing the computer that the sequential file will hold a certain number of records of a particular length.
B. instructing the computer the program will only hold a certain number of records.
C. placed with the DATA and READ statements at the end of a file.
D. placed at the end of the OPEN command.

——— 5. This program line:

1000 PRINT CHR$(4);"WRITE TELS, R";N

would

A. precede the OPEN command and designate the RECORD you wish to use.
B. print:"WRITE TELS, R";N once an "N" was entered.
C. follow the OPEN command and designate record you wish to use.
D. follow the CLOSE command and designate the record that was written.

Assignment For Lesson Thirteen

Your assignment is to write a program which writes data to and reads data from a random access data file.

You may place the WRITE and READ operations in separate programs or you may combine the two operations as different modules of the same program.

You may wish to write a separate program to initialize the record counter in the zero record, or control record, of the data file.

Remember to WRITE the new number of records to the control record as part of the ending section of the program.

Additional Assignment For Lesson Thirteen, #1

Write a program which will allow the operator to enter the names of ten classes he or she takes in school, the beginning time of each class, and the ending time of each class. Store the data in a sequential file named "SCHEDULE".

Allow the operator to enter, view or change the class schedule.

Your program will need to give the operator three options: (1) enter schedule, (2) change schedule and (3) view schedule.

Each operation will be a separate module in your program.

Place the data in a two-dimensional array as: DIM C$(10,3). Each row is a separate class. Element one of each row is for the description. Element two of each row is for the beginning time. Element three is for the ending time. (The program need not use the zero elements.) Use a FOR-NEXT loop when writing data to and reading data from the file.

Additional Assignment For Lesson Thirteen, #2

Write a program which uses a random access file to allow the operator to enter the names and descriptions of the programs on his or her diskette.

Provide add, change and display capabilities, each in a separate program module.

Use record number zero to store the number of the last record written.

You may want to write a small, separate program to open the file and write the beginning value of the record counter to the file.

LESSON FOURTEEN:

MORE PROGRAMMING TIPS AND TECHNIQUES

LESSON OBJECTIVES

Lesson Fourteen concludes the course by introducing you to additional programming techniques which you may wish to use or which you may encounter as you study BASIC programs listed in other books or magazines.

The first portion of the lesson is designed to assist you in interpreting programs listed in other computer materials. You will learn to recognize several other commands which you may encounter. You will learn another screen formatting technique and how to recognize a machine language subroutine incorporated into a BASIC program.

The next portion of the lesson teaches you the editing and debugging commands of the computer.

Finally, a sophisticated program utilizing the commands you have learned in this course is presented as an example of the type of programming you are ready to learn next.

The new BASIC commands introduced in this lesson are: **GET, RESTORE, CALL, TRACE** and **NOTRACE.** You are also introduced to the **ESC** editing commands.

The "DECISION" program is an example of the pure logic capacities of the computer. In this program the operator is assisted in making a decision. The decision can be one of two types: selecting an item from many alternatives (up to eight items can be considered against each other in this particular program) or choosing a course of action from a number of possible courses (this program allows up to eight courses of action to be considered).

As with any program, the effectiveness of the "DECISION" program is limited only by the input of the operator. In order for the program to work at its maximum efficiency, the operator must *carefully* evaluate which choice is most important to him, since all other choices are weighed against that choice as a constant.

Program Listing For Lesson 14

Screen Poke Positions

```
10  REM :: SCREEN POKE POSITIONS
20  REM   SAVE SCRN POKE
100 W$ = "****************************************"
110 S$ = "::::::::::::::::::::::::::::::::::::::::"
200  REM **********************
210  REM :: WINDOW POKES
220  REM :: LEFT/POKE 32,<0-39>
230  REM :: WIDTH/ "  33,<1-39>
240  REM :: TOP/    "  34,<0-23>
250  REM :: BOTTOM "  35,<0-23>
260  REM :: TEXT/ RESETS WINDOW
270  REM **********************
1000  TEXT : HOME
1010  PRINT "ENTER SCROLLING WINDOW DATA"
1020  INPUT "ENTER LEFT POSITION (0-39): ";L1
1030  PRINT "ENTER WIDTH OF WINDOW (1-";40 - L1;: INPUT ") ";W1
1040  INPUT "ENTER TOP POSITION (0-23) ";T1
1050  PRINT "ENTER BOTTOM (";T1 + 1;: INPUT "-23) ";B1
```

```
1060  HOME : FOR I = 1 TO 24: PRINT S$: NEXT I
1070  POKE 32,L1: REM  LEFT
1080  POKE 33,W1: REM  WIDTH
1090  POKE 34,T1: REM  TOP
1100  POKE 35,B1: REM  BOTTOM
1110  HOME
1120  SPEED= 220
1200  FOR I = 1 TO 30: PRINT W$: NEXT I
1210  TEXT : VTAB 22: HTAB 1: SPEED= 255
1220  PRINT "PRESS 'E' TO END OR"
1230  VTAB 23: HTAB 1
1240  PRINT "ANY OTHER KEY TO CONTINUE ";
1250  GET E$: PRINT
1260  IF E$ <  > "E" THEN 1000
9000  HOME : END
```

Music Machine Language Subroutine

```
10  REM :: MUSIC M/L SUBR
20  REM   SAVE TONES
100  DATA 173,48,192,136,208,5,206,1,3,240,9,202,208,245,
    174,0,3,76,2,3,96
200  FOR AD = 770 TO 790
210  READ N: POKE AD,N
220  NEXT AD
1000  FOR MUSIC = 100 TO 10 STEP  - 1
1010  POKE 768,MUSIC: REM TONE
1020  POKE 769,15: REM  DURATION
1030  CALL 770: REM M/L SUBR
1040  NEXT MUSIC
9000  END
```

Hello

```
10  REM :: VIDEO SOFT HELLO PGM
20  REM   SAVE HELLO
1000  HOME
1010  PRINT "VIDEO SOFT DISKETTE"
1020  PRINT "BENJAMIN MCCORKLE"
1030  PRINT  CHR$ (4);"CATALOG"
```

Trace

```
10  REM :: LINE NUMBER TRACE
20  REM   SAVE TRACE
1000  TRACE
1010  FOR I = 1 TO 10
1020  PRINT  CHR$ (7);
1030  NEXT I
1040  NOTRACE
1050  END
```

```
10  REM :: DECISION MAKER PGM ::
20  REM  SAVE DECISION
30  REM ::::::::::::::::::::::
40  REM :: HOUSEKEEPING SECTION
50  DIM F$(10),A$(8),F(10),A(8),E(8,10),C(8)
100  DATA  2,10
110  READ D1,L1
120 Z$ = "000000000000"
130 B$ = "            "
200  DATA 2,4,9,13,1,8,14,10
210  FOR I = 1 TO 8: READ C(I): NEXT I
980  REM ::::::::::::::::::::::
990  REM :: TITLE PAGE
1000  HOME : INVERSE : VTAB 5: HTAB 10
1010  PRINT " D E C I S I O N ": HTAB 13
1020  PRINT " M A K E R ": NORMAL : VTAB 8
1030  PRINT "THIS PROGRAM HELPS YOU MAKE"
1040  PRINT "DECISIONS BY ORGANIZING YOUR"
1050  PRINT "OPINIONS AND FACTS."
1060  PRINT : PRINT "WHICH OF THESE BEST DESCRIBES"
1070  PRINT "THE DECISION YOU HAVE TO MAKE?": PRINT
1080  PRINT " 1 - ONE ITEM FROM MANY."
1090  PRINT " 2 - A COURSE OF ACTION."
1100  VTAB 19
1110  INPUT "WHICH TYPE? 1 OR 2 (0 TO END): ";C
1120  ON C + 1 GOTO 9000,1200,1140
1130  GOTO 1100
1140 S$ = "ACTION"
1150  GOTO 1300
1200  HOME : VTAB 18
1210  PRINT "ENTER THE TYPE OF ITEM YOU"
1220  INPUT "NEED TO DECIDE UPON: ";S$
1280  REM ::::::::::::::::::::::
1290  REM :: CONTROL SECTION
1300  GOSUB 2000: REM ALTERNATIVES
1400  GOSUB 2500: REM FACTORS
1500  GOSUB 3000: REM RANK FACTOR
1600  GOSUB 4000: REM RATE ALTERNATIVE
1700  GOSUB 5000: REM BEST ALTERNATIVE
1800  GOSUB 6000: REM PRNT ALTERNATIVE
1900  GOSUB 7000: REM GRAPH IT
1950  GOTO 1000
1980  REM ::::::::::::::::::::::
1990  REM :: ALTERNATIVES SUBR
2000  HOME
2010  PRINT "YOU MUST FIRST ENTER A LIST OF"
2020  PRINT "EACH ";S$;" YOU ARE CONSIDERING."
2030  PRINT : PRINT "THE ORDER IS OF NO IMPORTANCE NOW."
2040  PRINT "YOU MAY ENTER UP TO 8 ";S$;"S."
2050  VTAB 21: PRINT "(AFTER LAST ENTRY, PRESS RETURN)"
2060  PRINT "(INSTEAD OF ADDING TO THE LIST.)"
2070  VTAB 7
2080 A0 = 0
2100 A0 = A0 + 1
2110  PRINT "NUMBER ";A0;: INPUT ": ";A$(A0)
2120  IF A$(A0) = "" THEN 2200
```

```
2130  IF A0 < 8 THEN 2100
2140  GOTO 2210
2200 A0 = A0 - 1
2210  HOME : PRINT "OK. HERE'S A LIST OF THE ";A0;" ";S$;"S"
2220  PRINT : FOR I = 1 TO A0: PRINT A$(I): NEXT I
2230  VTAB 18: INPUT "CORRECT? (Y/N) ";AN$
2240  IF AN$ = "N" THEN 2000
2250  IF AN$ <  > "Y" THEN 2230
2260  RETURN
2480  REM ::::::::::::::::::::::
2490  REM :: FACTORS SUBR
2500 F0 = 0
2510  HOME : PRINT "NOW YOU MUST DETERMINE WHICH FACTORS"
2520  PRINT "ARE IMPORTANT TO YOUR DECISION"
2530  PRINT "YOU MAY ENTER UP TO 10"
2540  PRINT : PRINT "(AGAIN, THE ORDER IS OF NO IMPORTANCE.)"
2550  VTAB 21: PRINT "(AFTER LAST FACTOR, PRESS RETURN)"
2560  PRINT "(INSTEAD OF ADDING TO THE LIST. )"
2570  VTAB 7
2600 F0 = F0 + 1
2610  PRINT "FACTOR ";F0;: INPUT ": ";F$(F0)
2620  IF F$(F0) = "" THEN 2690
2630  IF F0 < 10 THEN 2600
2640  GOTO 2700
2690 F0 = F0 - 1
2700  HOME : VTAB 3
2710  PRINT "THIS IS THE LIST OF THE ";F0;" FACTORS:"
2720  PRINT
2730  FOR I = 1 TO F0: PRINT I;": ";F$(I): NEXT I
2740  VTAB 20: INPUT "ARE THEY CORRECT (Y/N): ";AN$
2750  IF AN$ = "N" THEN 2500
2760  IF AN$ <  > "Y" THEN 2740
2770  RETURN
2980  REM ::::::::::::::::::::::
2990  REM :: RANK THE FACTORS
3000  HOME : PRINT "NOW, YOU MUST DETERMINE THE"
3010  PRINT "RELATIVE IMPORTANCE OF THE FACTORS"
3020  PRINT "ON A SCALE FROM 0 TO 100"
3030 X =  INT (F0 *  RND (1) + 1):F(X) = 50: VTAB 5
3040  PRINT "THE COMPUTER HAS ASSIGNED A"
3050  PRINT "VALUE OF 50 TO ";F$(X);"."
3060  VTAB 11: PRINT "IF ";: INVERSE : PRINT F$(X);: NORMAL
3070  PRINT " IS WORTH 50,"
3100  FOR I = 1 TO F0: IF I = X THEN 3160
3110  VTAB 14: PRINT  TAB( 40);: VTAB 14: HTAB 1
3120  PRINT "RATE ";: INVERSE : PRINT F$(I);: NORMAL : PRINT ":"
3130  VTAB 16: PRINT  TAB( 40);: VTAB 16: HTAB 1
3140  VTAB 16: INPUT "(MORE, LESS, THE SAME?): ";F(I)
3150  IF F(I) < 0 OR F(I) > 100 THEN 3140
3160  NEXT I
3200  RETURN
3980  REM ::::::::::::::::::::::
3990  REM :: RATE ALTERNATIVES SUBR
4000  HOME : VTAB 4
4010  PRINT "THE NEXT STEP IN THE PROCESS"
4020  PRINT "IS TO RATE EACH ";S$;" ACCORDING"
4030  PRINT "TO FACTORS YOU SAID WERE IMPORTANT."
4040  PRINT : PRINT "THE COMPUTER WILL ASSIGN A VALUE"
```

```
4050  PRINT "OF 50 TO ONE OF THE ";S$;"S."
4060  PRINT : PRINT "ALL OTHER ";S$;"S WILL BE GIVEN"
4070  PRINT "A RATING HIGHER OR LOWER THAN 50"
4080  PRINT "DEPENDING ON HOW MUCH BETTER"
4090  PRINT "OR WORSE YOU THINK IT IS."
4100  VTAB 18: INPUT "(PRESS RETURN TO BEGIN) ";A$
4480  REM ::::::::::::::::::::::
4490  REM :: RATING LOOP ::
4500  FOR I = 1 TO F0: HOME : VTAB 4
4510  PRINT "CONSIDERING ONLY THE FACTOR OF"
4520  INVERSE : HTAB 10: PRINT F$(I): NORMAL
4530  PRINT "RATE EACH ";S$;"."
4540 X =  INT (A0 *  RND (1) + 1):E(X,I) = 50
4550  VTAB 10: PRINT "CONSIDERING THAT THE COMPUTER HAS RATED"
4560  INVERSE : PRINT A$(X);: NORMAL : PRINT " AT 50
4600  FOR J = 1 TO A0: IF J = X THEN 4700
4610  VTAB 13: PRINT  TAB( 40): VTAB 13: HTAB 1
4620  PRINT "HOW WOULD YOU RATE ";
4630  INVERSE : PRINT A$(J);: NORMAL : PRINT "?"
4640  VTAB 14: PRINT  TAB( 40): HTAB 1
4650  VTAB 14: INPUT "(MORE, LESS, THE SAME?): ";E(J,I)
4660  IF E(J,I) < 0 OR E(J,I) > 100 THEN 4650
4700  NEXT J
4710  NEXT I
4720  RETURN
4980  REM ::::::::::::::::::::::
4990  REM :: BEST ALTERNATIVE SUBR
5000  FOR J = 1 TO A0:A(J) = 0
5010  FOR I = 1 TO F0:A(J) = A(J) + E(J,I) * F(I)
5020  NEXT I: NEXT J
5280  REM ::::::::::::::::::::::
5290  REM :: SORT ALTERNATIVES
5300 SF = 0
5310  FOR I = 1 TO A0 - 1
5320  IF A(I) >  = A(I + 1) THEN 5360
5330 TS = A(I):A(I) = A(I + 1):A(I + 1) = TS
5340 TS$ = A$(I):A$(I) = A$(I + 1):A$(I + 1) = TS$
5350 SF = 1
5360  NEXT I
5370  IF SF = 1 THEN 5300
5480  REM ::::::::::::::::::::::
5490  REM :: FINAL RANKINGS
5500 C = 100 / A(1)
5510 A(1) = 100
5520  FOR J = 2 TO A0
5530 A(J) = C * A(J)
5540  NEXT J
5550 D = A(1) - A(2)
5560  RETURN
5980  REM ::::::::::::::::::::::
5990  REM :: PRINT SUBR
6000  HOME : VTAB 4: PRINT "WELL, YOU BEST CHOICE SEEMS TO BE:"
6010  FLASH : HTAB 15: PRINT A$(1): NORMAL
6020  IF D < 11 THEN  PRINT "BUT IT'S PRETTY CLOSE."
6030  IF D > 10 AND D < 21 THEN  PRINT "BY A FAIR MARGIN."
6040  IF D > 20 THEN  PRINT "BY A GOOD-SIZED MARGIN."
6050  VTAB 10: INVERSE
6060  PRINT " ";S$; TAB( 28);"RANKING ": NORMAL
```

```
6070  FOR I = 1 TO A0
6080 N = A(I): GOSUB 8000
6090  PRINT A$(I); TAB( 25);N$
6100  NEXT I
6120  VTAB 22: INPUT "PRESS RETURN FOR BAR GRAPH. ";A$
6130  RETURN
6980  REM ::::::::::::::::::::::
6990  REM :: GRAPH SUBR
7000  HOME : GR :H =  INT (40 / A0)
7010  FOR I = 1 TO A0
7020  VTAB 21
7030 H1 = (I - 1) * H + 1
7040  HTAB H1: PRINT  LEFT$ (A$(I),H - 1);
7050 N$ =  STR$ (A(I))
7060  VTAB 22: HTAB H1
7070  PRINT  LEFT$ (N$,4);
7100 X = 39 - A(I) / 100 * 39
7110  COLOR= C(I)
7120 Y = H1 - 1:Z = Y + H - 1
7130  FOR J = Y TO Z
7140  VLIN X,39 AT J
7150  NEXT J
7160  NEXT I
7170  PRINT
7180  INPUT "(PRESS RETURN TO END) ";A$
7190  TEXT
7200  RETURN
7980  REM ::::::::::::::::::::::
7990  REM :: NUMBER FORMAT SUBR
8000 N$ =  STR$ (N):L2 =  LEN (N$)
8010  FOR L = 1 TO L2
8020  IF  MID$ (N$,L,1) = "." THEN 8050
8030  NEXT L
8040 N$ = N$ + "." +  RIGHT$ (Z$,D1): GOTO 8080
8050 ND = L2 - L
8060  IF ND > D1 THEN N$ =  LEFT$ (N$,L + D1): GOTO 8080
8070  IF ND < D1 THEN AD = D1 - ND:N$ = N$ +  RIGHT$ (Z$,AD)
8080 L2 =  LEN (N$)
8090  IF L2 < L1 THEN AB = L1 - L2:N$ =  RIGHT$ (B$,AB) + N$
8100  RETURN
8980  REM ::::::::::::::::::::::
8990  REM :: END OF JOB
9000  TEXT : HOME
9010  VTAB 8: HTAB 10
9020  PRINT "THE BEST DECISION WAS"
9030  VTAB 10: INVERSE
9040 C = 20 -  LEN (A$(1)) / 2
9050  HTAB C: PRINT A$(1)
9060  NORMAL : VTAB 14: HTAB 13
9070  PRINT "PROGRAM ENDED"
9080  VTAB 22
9090  END
```

Lesson Fourteen Self-Check Test #1

Directions — Choose the answer from the bottom of the page that BEST defines the statement. Place the letter of your choice in the space provided. Choices may be used more than once.

——— 1. The name of a type of display used to give the operator a choice among several options.

——— 2. A command that allows the operator to enter only one character and not have to press RETURN.

——— 3. The portion of the screen in which a regular printed message will be displayed.

——— 4. "POKE 32" and the position 0-39 will set this side of your scrolling window.

——— 5. "POKE 33" and the number of characters to be displayed sets this part of your scrolling window.

——— 6. "POKE 34" and the line number 0-23 will set this side of your scrolling window.

——— 7. "POKE 35" and the line number which must be greater than the top line will set this side of your scrolling window.

——— 8. Instructs the computer to stop displaying line numbers.

——— 9. This command often follows the instruction: PRESS ANY KEY TO CONTINUE

——— 10. A command associated with DATA and READ commands which allows you to read from the beginning of the list in the DATA statement more than once.

——— 11. This command places a machine language subroutine directly into the memory of the computer.

——— 12. This command executes a machine language subroutine beginning at the address specified by the numbers following it.

——— 13. An "X" typed with this key allows you to cancel a program line or command.

——— 14. This command instructs the computer to display the line numbers as it executes the program.

A . BOTTOM LINE
B . CALL
C . CONTROL
D . COPY A
E . GET
F . LEFT SIDE
G . MENU
H . NOTRACE
I . POKE
J . RENUMBER
K . RESTORE
L . SCROLLING WINDOW
M . TOP LINE
N . TRACE
O . WIDTH

Lesson Fourteen Self-Check Test #2

Directions — Choose the answer that BEST completes the statement. Write the letter of your choice in the space provided.

——— 1. To enter editing mode, press the

A. ESC key.
B. CTRL key.
C. RETURN key.
D. REPT key.

——— 2. To move the cursor up the screen, press ESC followed by the

A. I key.
B. J key.
C. K key.
D. M key.

——— 3. To move the cursor to the left, press ESC followed by the

A. I key.
B. J key.
C. K key.
D. M key.

——— 4. Before changing a line, use ESC followed by the I, J, K, M keys to position the cursor at

A. the portion of the line to be changed.
B. the space following the line number.
C. the end of the line to be changed.
D. the first digit of the line number.

——— 5. To leave edit mode and change a program line, press the

A. SHIFT key.
B. CTRL key.
C. Space Bar.
D. REPT key.

——— 6. To include a portion of the line to be changed, use the

A. J key.
B. K key.
C. Right Arrow key.
D. Left Arrow key.

——— 7. To erase from the cursor position to the end of the screen line, use

A. CTRL-X
B. ESC-E
C. ESC-F
D. ESC-@

——— 8. To erase from the cursor position to the bottom of the screen, use

A. CTRL-X
B. ESC-E
C. ESC-F
D. ESC-@

Lesson Fourteen Self-Check Test #3

Directions — The following is a series of steps you must follow to LOAD a program using the ESC keys and a CATALOG listing. Put them in the correct order by writing the letter of your choice in the numbered spaces provided.

1. ——— A. Press the Space Bar to leave edit mode.

2. ——— B. Use the Right Arrow key to include the program file name.

3. ——— C. Press the RETURN key.

4. ——— D. Type LOAD and a space.

5. ——— E. Use ESC-I to position the cursor to the front of the program to be LOADed.

Directions — The following statements are a series of steps to be followed to SAVE a program using the ESC keys and the SAVE REM in line 20. Put them in the correct order by writing the letter of your choice in the numbered spaces provided.

6. ——— A. Use ESC-I and J to position the cursor to the beginning of line number 20.

7. ——— B. LIST 20.

8. ——— C. Use the Right Arrow key to include the SAVE command and the program file name.

9. ——— D. Use the Space Bar to erase the line number and the REM.

10. ——— E. Press the RETURN key.

11. ——— F. Press the Space Bar to leave edit mode.

Appendices

Answers To Self-Check Tests

LESSON 1
TEST 1

1. O
2. G
3. F
4. N
5. J or D
6. B
7. L
8. E
9. I
10. H
11. K
12. C
13. A

LESSON 1
TEST 2

1. B
2. H
3. A
4. F
5. G
6. C
7. A
8. D
9. E,F
10. E

LESSON 1
TEST 3

1. F QUOTATION MARKS
2. F NOT INTERCHANGEABLE
3. T
4. T
5. T
6. F PERMANENT
7. T
8. F FLOPPY
9. T
10. T

LESSON 1
TEST 4

1. BASIC
2. INIT
3. LIST
4. DIRECT
5. HOME
6. NEW
7. PRINT
8. HARD DISK
9. RUN
10. STATEMENT LINE
11. GREETING PROGRAM
12. BOOT

LESSON 2
TEST 1

1. I
2. Q
3. A
4. G
5. L
6. H
7. K
8. M
9. D
10. E
11. J
12. F
13. N
14. C

LESSON 2
TEST 2

1. T
2. T
3. T
4. T
5. F NORMAL
6. T
7. F PRINT
8. T
9. F NOT ALWAYS NECESSARY
10. F FILE NAME
11. T
12. T
13. F SOUND THE BELL

LESSON 2
TEST 3

1. C
2. B
3. A
4. D
5. B
6. C
7. D
8. D
9. A
10. E
11. B
12. C
13. A
14. C
15. B
16. E

LESSON 3
TEST 1

1. D
2. B
3. C
4. A
5. A
6. A
7. D
8. C
9. B
10. D
11. D
12. D
13. D

LESSON 3
TEST 2

1. O
2. B
3. E
4. M
5. C
6. L
7. I
8. J
9. K
10. L
11. D
12. E, OR F
13. H
14. N
15. G
16. Q
17. A
18. E, OR F

LESSON 4
TEST 1
Part 1

ADDITION (+)
MULTIPLICATION (*)
DIVISION (/)
SUBTRACTION (-)
EXPONENTIATION (^)

Part 2

4
1
2
3

Part 3

(1)
N = 4 + 8^2/8-1
= 4 + 64/8-1
= 4 + 8-1
N = 11

(2)
N = (24/6 + 3)/(3 + 4)
= (4 + 3)/7
= 7/7
N = 1

(3)
N = (5*2 + (6-2))/7 + 3^3 + 2
= (10 + 4)/7 + 27 + 2
14/7 + 27 + 2
= 2 + 27 + 2
N = 31

LESSON 4
TEST 2

1. T
2. F MULTIPLY
3. T
4. T
5. F LEFT TO RIGHT
6. T
7. T
8. F 24
9. F WRAP AROUND THE SCREEN AND CONTINUE TO PRINT
10. T
11. T
12. T
13. F 132
14. F LINE 1, POSITION 1

LESSON 4
TEST 3

1. A
2. F
3. H
4. A
5. B,G
6. A
7. G
8. F
9. B,H
10. G
11. B
12. F
13. H
14. G
15. B OR H
16. G
17. G

LESSON 5
TEST 1

1. GOTO
2. CONTROL-C
3. IF-THEN
4. PROMPTED INPUT or INPUT
5. IF-THEN
6. GOTO
7. AND
8. OR
9. CONTROL-RESET
10. ETERNAL LOOP

LESSON 5
TEST 2

1. =
2. <
3. >
4. < >

LESSON 5
TEST 3

1. CIRCLE PRINT "YES"
2.
3. CIRCLE PRINT "YES"
4.
5. CIRCLE PRINT "YES"
6.
7.
8.
9. CIRCLE PRINT "YES"
10. CIRCLE PRINT "YES"

LESSON 6
TEST 1

1. J
2. F
3. I
4. H
5. D
6. L
7. E
8. A
9. C
10. G

LESSON 6
TEST 2

T
2. F TRUE
3. F RUN
4. T
5. T
6. T
7. T
8. F BEGINNING, ENDING
9. T
10. T
11. F ENDED

LESSON 7
TEST 1

1. B or C
2. A
3. B
4. C
5. I
6. H
7. E
8. A
9. B
D
11. F
12. H
13. B

LESSON 7
TEST 2

1. A
2. C
3. B
4. D
5. C
6. E
7. D
8. C

LESSON 8
TEST 1

1. SUBROUTINE
2. GOSUB
3. FALL THROUGH
4. ON...GOTO
5. FLOWCHARTING
6. FLOWCHARTING TEMPLATE
7. RETURN
8. REM
9. MAIN SECTION or CONTROL SECTION
10. TERMINAL or OVAL
11. DECISION
12. GOSUB
13. RETURN

LESSON 8
TEST 2

1. C
2. A
3. E
4. E
5. A
6. D
7. C
8. A
9. B
10. E
11. A
12. B
13. C
14. D

LESSON 8
TEST 3

1. D
2. B
3. C
4. C
5. A

LESSON 9
TEST 1

1. F NAME
2. T
3. F AN INDIVIDUAL VARIABLE
4. F SUBSCRIPTING
5. T
6. F COMMA
7. T
8. F DATA AND READ
9. T
10. T
11. F TWO
12. T
13. T
14. F 11

LESSON 9
TEST 2

1. S
2. G
3. N
4. D
5. I
6. A
7. C
8. B
9. I
10. L
11. K
12. H
13. N
14. R

LESSON 10
TEST 1

1. I
2. H
3. B
4. C
5. E
6. E
7. A
8. J
9. G
10. E

LESSON 10
TEST 2

1. D
2. A
3. C
4. B
5. A
6. C
7. B
8. D
9. C
10. B

LESSON 11
TEST 1

1. T
2. A
3. D
4. E
5. Q
6. F
7. B
8. I
9. O
10. M
11. U
12. P
13. R
14. K
15. H

LESSON 11
TEST 2

1. E
2. B
3. D
4. D
5. B
6. A
7. C
8. C
9. D
10. B
11. C

LESSON 12
TEST 1

1. C
2. L
3. J
4. I
5. N
6. K
7. E
8. M
9. A
10. G
11. B
12. H
13. G
14. G

LESSON 12
TEST 2

1. C
2. A
3. D
4. C
5. B
6. B
7. A
8. B
9. D
10. A

LESSON 12
TEST 3

1. B
2. D
3. A
4. C

LESSON 13
TEST 1

1. F PERMANENTLY
2. T
3. T
4. F T
5. T
6. T
7. T
8. T
9. F ALWAYS

LESSON 13
TEST 2

1. J
2. C
3. I
4. E
5. B
6. K
7. D
8. I
9. A
10. H

LESSON 13
TEST 3

1. B
2. D
3. A
4. D
5. C

LESSON 14
TEST 1

1. G
2. E
3. L
4. F
5. O
6. M
7. A
8. H
9. E
10. K
11. I
12. B
13. C
14. N

LESSON 14
TEST 2

1. A
2. A
3. B
4. D
5. C
6. C
7. B
8. C

LESSON 14
TEST 3

1. D OR E
2. E OR D
3. A
4. B
5. C
6. B
7. A
8. F
9. D
10. C
11. E

SCIENTIFIC NOTATION OR E-NOTATION

The APPLE computer will display only a maximum of nine digits per number. If the computer is instructed to print a number larger than nine digits, it will convert the number into E-Notation.

For example, if the computer is instructed to print the ten-digit number, 1762084930, it will convert the number to E-Notation before displaying it on the CRT.

The computer first rounds the ten-digit number to a nine-digit number. Numbers are rounded by the computer if they have more than nine digits. Any number that has fewer than ten digits will not be rounded.

10 Digit Number	**E-Notation**
1762084930	1.76208493E+09

In this example, the E+09 means that the decimal point actually belongs nine places to the right. The computer will display any number larger than nine digits in Scientific or E-Notation.

---

12 Digit Number	**E-Notation**
348957000000	3.48957E+11

In this example, first note that the computer did not print the trailing zeros. The APPLE does not print zeros that are at the end of the number, to the right of the decimal; nor does it print zeros that are at the beginning of the number, to the left of the decimal. The E+11 means that the decimal belongs eleven places to the right.

The APPLE also displays very small decimal numbers in E-Notation.

11 Digit Decimal Number	**E-Notation**
0.00000005312	5.312E-08

In this example, the computer does not print the zero to the left of the decimal point. The computer converts the very small decimal number to E-Notation. E-08 means the decimal point belongs eight places to the left.

You will very rarely use E-Notation when writing programs or when entering numbers into the computer. However, since the computer displays very small numbers and numbers larger than nine digits in E-Notation, it is important to be able to recognize the structure of E-Notation.

GLOSSARY

Accumulator - a variable used to keep track of a running total.

Address - a specific location in the computer's memory.

Animation - creates the illusion of motion on the CRT by drawing an object, erasing the object, and then redrawing the object in a different location.

Array - a set of variables that are linked together by the way they are named. All variables within the same array have a common name and are numbered, or subscripted, to tell them apart.

Array Element - a single subscripted variable within an array.

Assembly Language - a programming language in which letters are substituted for numbers when giving instructions to the processor.

Asterisk - the symbol for multiplication in BASIC.

Base Zero Machine - refers to a computer which begins the subscripting sequence at "zero" instead of "one".

BASIC - a high-level computer language; an acronym for "Beginner's All-Purpose Symbolic Instruction Code".

Boot - the act of adding the Disk Operating System (DOS) to the language in which the computer will be programmed. (See "DOS".)

Branching Gosub - used as an alternate term for the "ON...GOSUB" command. It is also referred to as a "Computed GOSUB". (See "ON...GOSUB"explanation listed in the Summary of Commands Appendix.)

Branching Goto - used as an alternate term for the "ON...GOTO" command. It is also referred to as a "COMPUTED GOTO". (See "ON...GOTO" explanation listed in the Summary of Commands Appendix.)

Bug - an error in a program.

Byte - the representation of a single character of information. For example, one sector on a diskette can contain up to 256 bytes of information. Each memory location in the computer can store one byte of information. (See "Sector" and "Character".)

Caret (^) - the symbol for exponentiation in BASIC.

Character - refers to any letter, digit, punctuation mark and the space.

Character String - A series of adjacent letters, numbers, symbols or spaces.

Character Variable - (or string variable) a variable which stores a character string. A character variable name is identified by a dollar sign at the end.

Chip - an integrated electronic semi-conductor used in electronic circuits.

COBOL - a high-level computer language; an acronym for COmmon Business Oriented Language.

Command - an instruction to the computer.

Computed Gosub - See "Branching Gosub".

Computed Goto - See "Branching Goto."

Computer - a machine that can be given a set of instructions (a program), store the instructions, and execute the instructions.

Concatenation - refers to putting together two or more strings or pieces of strings using the plus ("+") sign in between.

Control Record - a record in a random access file reserved by the programmer to store information describing the file itself. Such information usually includes the number of records in the file. Usually, the number zero record is reserved for this use.

CRT - cathode ray tube; a computer's display screen.

Cursor - the flashing block of light visible on the CRT.

Data - information of any kind.

Data File - a diskette file used for the permanent storage of information. Also called "Text File".

Debug - to find and correct the errors in a program.

Deferred Execution - execution of program statements when the computer is instructed to RUN a program; requires a line number. Also called "Delayed Execution". (See "Direct Command".)

Delay Loop - See "Timing Loop".

Direct Command - immediate execution of a command by a computer, without the need of a program. Also called "Immediate Execution". (See "Deferred Execution".)

Diskette - similar to magnetic tape. Information is recorded and then permanently stored on diskettes (until it is erased). Also called a "Disk" or a "Floppy".

Diskette Drive - a mechanical device that the computer uses for holding and processing diskettes.

Display - See "CRT".

DOS - stands for "Disk Operating System". A high-level specialized language used for controlling the diskette and diskette operations.

Dummy Input - an INPUT statement used to temporarily stop the program until the operator is ready to proceed.

Error Trap - a program routine used to keep a program from stopping when an error is encountered. (See "ONERR GOTO" in the Summary of Commands Appendix.)

Execute - what the computer does when it runs a program, or a portion of a program, or when the computer performs a command or statement.

File - information stored on a diskette under a single file name.

File Name - the name of a diskette file.

File Pointer - an invisible mark used by the computer to keep its place in a data file.

Flashing Mode - the mode in which the computer alternates the print on the CRT from INVERSE to NORMAL.

Floppy - another name for a diskette; called a "Floppy" because it is flexible. (See "Diskette".)

Flowcharting - a program design technique using symbols to represent program operations, data, flow, etc. (See "Program Flow".)

Formatting - the process of controlling the printing of information and the position of the cursor on the CRT.

FORTRAN - a high-level computer language; stands for FORmula TRANslation.

Greeting Program - a program that will run each time that the Disk Operating System (DOS) is booted. (See "Boot".)

High-Level Language - a programming language which groups instructions together into understandable words. Examples are BASIC, COBOL, FORTRAN and PASCAL.

High Res Screen - mode used in high resolution graphics. The screen contains 280 by 192 plotting points.

I/O - stands for input/output operations. The INPUT command is an example of an input operation in which data is received from the keyboard. The PRINT command is an example of an output operation in which data is displayed on the CRT.

Initialization - the process by which a blank diskette is marked and organized into tracks and sectors. Upon completion of this process, the diskette is ready to store programs and data.

Integer - a whole number.

Inverse Mode - the mode in which the computer writes black letters on a white background on the CRT by using the INVERSE command within the computer program.

Line Number - numbers used within a program to put program statements in sequence.

Logic Paragraph - a technique used when designing a program; a short statement of the programming tasks to be performed in a given section of a computer program.

Loop - a repetition of program lines.

Loop Variable - the numeric variable used in a "FOR...NEXT"' statement.

Lo Res Screen - mode used in lo resolution graphics. The screen contains 40 by 40 plotting points with four text screen lines at the bottom.

Machine Language - a programming language which uses numbers to give instructions directly to the computer processor.

Memory - that portion of the computer where information is stored.

Menu - a displayed list of options from which the operator can choose.

Modular Programming - the technique of writing computer programs in sections, or modules, each of which executes a specific portion of the programming task.

Module - a set of instructions which perform a portion of a programming task. Also called "section".

Multi-Dimensional Array - functions as a set of arrays linked together under a common variable name. For example, a two-dimensional array contains two rows with each row containing an equivalent number of elements. (See "ARRAYS").

Multiple Input Statement - an INPUT statement using more than one variable, allowing the operator to enter more than one item. The variables must be separated by commas.

Multiple Statement Line - a programming line which contains more than one command. Multiple commands on a single line must be separated by colons.

Nested Loop - a loop, or series of loops, within a loop(s).

Normal Mode - the mode in which the computer normally operates on the CRT; white letters on a black background. When used within a program, the NORMAL command cancels the FLASH and INVERSE commands.

Null String - an empty character string or a string variable which has no value. For example, A$ = "". The length of a null string is zero.

Numeric Literal - an actual numeric value used within a program. It can contain the digits "zero" through "nine", a plus or minus sign, and a decimal point.

Numeric Variable - a variable which stores numeric values only. Unlike a character variable name, a numeric variable name does not use an ending dollar sign.

Paired Command - two commands which are normally used together. For example, FOR-NEXT; GOSUB-RETURN; DATA-READ.

PASCAL - a high level computer language; named after the famous 17th century French philosopher, mathematician and scientist Blaise Pascal.

Print Zones - a fixed beginning of a column of the CRT. (See "COMMA" explanation in the Summary of Commands Appendix.)

Processor - the electronic chip within a computer that interprets and executes instructions.

Program - a set of instructions that the computer will execute.

Program Flow - the sequence in which program statements are executed. (See "FLOWCHARTING".)

Program Outline - a technique used when designing a program; an outline of the modules comprising a computer program.

Prompt - a message on the screen that informs the computer operator that information is being requested and should be entered from the keyboard.

Prompted Input - prints a string literal prior to executing an INPUT command which requires an operator response.

Random Access File - a data file used by the computer to store and read sets of information in the order specified by the program.

Random Numbers - numbers without predetermined value such as a throw of dice.

Record - a space of defined length used for holding related information; is used with random access files.

Remark - comments and notes that are made within a program that help explain the program. The REM command precedes the remark.

Reserved Words - those words that are set aside to be used as specific commands to the computer.

Scientific Notation - translates a number into exponential form. Also called "E-Notation". (See SCIENTIFIC NOTATION Appendix.)

Screen - See "CRT".

Scrolling - the act of moving forwards or backwards through a list.

Scrolling Window - that portion of the CRT in which a regular printed message is displayed.

Section - See "Module".

Sector - a division of a track upon which information is physically stored on a diskette. (See "TRACKS".)

Sequential File - a data file used by the computer in which information is read and stored sequentially.

Statement Lines - a numbered set of computer instructions and/or commands within a program.

String Literal - any letter, number, symbol, or space placed within quotation marks.

String Variable - See "Character Variable".

Subroutine - a series of program lines which perform a specific task and can be accessed from anywhere in the program.

Subscript - a numeric value placed between parentheses following a variable name; used to identify an array element.

Subscripted Variable - a variable with a subscript. (See "Subscript".)

Syntax - rules that govern the structure of statement lines.

Syntax Error - a message issued by the computer telling the user that the proper language to give a command was not used.

Text File - See "Data File".

Text Screen - the APPLE display screen which is divided into 24 lines and 40 columns.

Text Window - the area in which text is displayed upon the CRT.

Timing Loop - a loop used specifically to stop the flow of a program for a given time.

Tracks - the bands or zones upon which information is physically stored on diskette.

Variable - a place in the memory of the computer in which information is stored and labeled with a unique name which may be used within a BASIC program.

Variable Name - a label used to identify a particular variable which must begin with a letter of the alphabet. While the name can be up to 238 characters long, the computer will only recognize the first two characters.

SUMMARY OF COMMANDS

The following summary of commands briefly describes the BASIC and DOS commands covered in the VideoSoft™ Instructional Programming Series For Applesoft™ BASIC

AND - establishes two or more conditions to be met in the "IF" portion of an "IF...THEN" statement. If all conditions are met, the command(s) following the "THEN" portion of the command will be executed.

APPEND - instructs the computer to open an existing sequential file for additional information.

ARROW KEYS - ← → - there are two such keys; one moves the cursor to the left and the other moves the cursor to the right. Neither affects the visual display. When the cursor is moved to the left, whatever it moves over when RETURN is pressed is erased. The right arrow key re-establishes each character the cursor moves over as part of the program line.

CALL - executes a machine language subroutine from a BASIC program.

CATALOG - a DOS command which instructs the computer to display a list of the files on a diskette upon the CRT.

CHR$(4) - instructs the computer that the next command is a DOS command.

CHR$(7) - instructs the computer to make its bell sound.

CLOSE - a DOS command which instructs the computer to prevent access to an open data file.

COLOR= - sets the color to be used in lo-resoluton graphics mode.

COMMA - when used within a PRINT statement, moves the cursor to the beginning of the next print zone.

CTRL C - can be used to interrupt the running or the listing of a program. When used, the computer will display the line number being executed when the program was interrupted.

CTRL RESET - can be used to interrupt the running or the listing of a program. (Experienced programmers, however, prefer "CTRL C".)

CTRL S - temporarily stops the listing of a program until another key is pressed.

CTRL X - instructs the computer to ignore any entry or edit on a line currently being typed.

DATA - begins a program statement which contains information to be placed into variables. (See "READ" command.)

DEL - deletes specified lines from a program.

DELETE - a DOS command which instructs the computer to permanently erase a program from the diskette.

DIM - shows the computer the largest subscripted variable to be used in an array.

END - instructs the computer to cease execution of program statements.

ESC @ - another way of performing the function of the HOME command.

ESC E - an editing function which erases the remainder of the program line being edited.

ESC F - an editing function which erases the CRT display from the position of the cursor to the end of the screen.

ESC I or **ESC J** or **ESC K** or **ESC M** - commands used to position the cursor at a specific location for editing a program without affecting the existing information. First press and release ESC key and then press I or J or K or M to position the cursor at the desired location. I moves the cursor up; J moves the cursor to the left; K moves the cursor to the right; and M moves the cursor down.

FLASH - sets the computer's screen to alternate from INVERSE to NORMAL print.

FOR...NEXT - the "FOR" establishes a loop by assigning a beginning and ending value to a variable. The "NEXT" command increases the value of the loop variable and sends the program back to the "FOR" command for the next pass of the loop.

GET - instructs the computer to place the value of a single character from the keyboard into a character variable. It does not display the character on the screen and it does not require the operator to press return.

GOSUB - instructs the computer to execute a subroutine which begins at a specified line number.

GOTO - instructs the computer to jump lines within a program to a specific line.

GR - sets the screen to lo-resolution graphics mode.

HLIN - draws a horizontal line in lo-resolution graphics mode from one specified point to another at a given row.

HOME - clears the text screen and moves the cursor to the upper left hand corner of the CRT.

HTAB - positions the cursor to a specified position on the 255 character print line.

IF...THEN - the "IF" part of the command sets up a condition to be met. If the condition is true, the computer will execute all instructions following the "THEN" part of the command. If the condition is false, then the computer will go to the next numbered statement line.

INIT - a DOS command which when followed by a program name, initializes a diskette by marking the tracks and sectors so that data and programs can be permanently recorded.

INPUT - assigns values from the keyboard to variables within a program.

INT ( ) - instructs the computer to remove the decimal portion of a positive number.

INVERSE - sets the computer's screen to print black letters on a white background.

LEFT$ ( ) - extracts a specified number of characters from the left-most portion of a character string.

LEN ( ) - instructs the computer to find the length of a string.

LET - used to assign values to variables.

LIST - displays the statement lines of the program currently stored in the computer's memory.

LOAD - A DOS command which instructs the computer to make a copy of the program file from a diskette and place it in the memory of the machine.

LOCK - a DOS command which instructs the computer to prevent a diskette file from being erased or altered.

MID$( ) - extracts a specified number of characters from any specified location within a character string.

NEW - clears the computer's program memory.

NORMAL - resets the computer's screen to print white letters on a black screen.

NOTRACE - instructs the computer to stop the execution of the TRACE command.

ONERR GOTO - instructs the computer to jump to a specific line if an error occurs.

ON...GOSUB - instructs the computer to branch to a subroutine based on the value of the variable used with "ON".

ON...GOTO - instructs the computer to branch to a line number based on the value of the variable used with "ON".

OPEN - a DOS command which instructs the computer to ready a data file for information storage and retrieval.

OR - establishes two or more conditions in the "IF" portion of an "IF...THEN" statement. If any of the conditions are met, the command(s) following the "THEN" portion of the statement will be executed.

PEEK - reads the numeric value at a specified address in the computer's memory.

PEEK (-16336) causes the speaker to click. The sound PEEK command is used to produce sound effects.

PEEK (-16384) reads the numeric code of the last key pressed on the keyboard. The keyboard PEEK is used in animation to accept input from the operator without stopping the flow of the program.

PLOT - instructs the computer to place a block of color at a designated row and column in lo-resolution graphics mode.

PLUS SIGN *(+)* - used to concatenate strings. (See "Concatenation" in Glossary.)

POKE - places a numeric value at a specified address in the computer's memory.

POKE 32,L - sets the left side of the scrolling window to the position specified by "L" in the range of 0 to 39.

POKE 33,W - sets the width of the scrolling window to "W". The width plus the left position must be less than 40.

POKE 34,T - sets the top of the scrolling window to "T" in the range 0 to 23.

POKE 35,B - sets the bottom of the scrolling window to "B" which must be greater than the top position and less than 24.

POKE -16368,0 - resets the computer keyboard after finding a specific key for which the program was testing. The keyboard POKE command is paired with the keyboard PEEK command.

PRINT - instructs the computer to display a value of a variable or a literal on the CRT.

READ - as a BASIC command, assigns the information from a DATA statement to specified variables.

READ - as a DOS command, instructs the computer to retrieve information from a data file.

REM - instructs the computer to ignore the statement line. It is used for explanatory remarks of the programmer.

RENAME - a DOS command which instructs the computer to change the name of the diskette file.

RESTORE - instructs the computer to return to the first data statement line within a program for use with the next READ command.

RETURN - is placed at the end of a subroutine to instruct the computer to return to the line containing the "GOSUB" command and start processing with the next statement.

RIGHT$ ( ) - extracts a specified number of characters from the right-most portion of a character string.

RND ( ) - instructs the computer to generate a random decimal number between zero and one.

RUN - a DOS command which instructs the computer to process the program it currently has stored in memory. RUN in conjunction with a program name will first load the program from diskette and then cause it to be executed.

SAVE - a DOS command which instructs the computer to make a copy of the program currently in memory within the computer and to store it on a diskette.

SCRN ( ) - instructs the computer to find the color of a specified location on the lo-resolution screen.

SEMICOLON - when used within a print statement, moves the cursor to the next space on the print line.

SPEED - controls the print rate of the CRT.

STEP - used with the "FOR...NEXT" commands to alter the amount the loop variable increases with each pass of the loop.

STR$ ( ) - gives a string representation of a numeric value.

TAB ( ) - prints a sufficient number of blanks to move the cursor from its current position to the position specified by the number within parentheses.

TEXT - sets the screen to text mode.

TRACE - instructs the computer to display the program line number as it executes that line.

UNLOCK - a DOS command which instructs the computer to remove the protection of the "LOCK" command.

VAL ( ) - finds the numeric value of a character string.

VLIN - draws a vertical line in lo-resolution graphics mode from one specified point to another at a given column.

VTAB - positions the cursor to any one of the twenty-four text screen lines.

WRITE - a DOS command which instructs the computer to store information in a data file.

RESERVED WORDS

This list contains all of the reserved words found in Applesoft BASIC, including those not covered in this course. Each reserved word takes up one byte of program storage.

&
ABS
AND
ASC
AT
ATN

CALL
CHR$
CLEAR
COLOR=
CONT
COS

DATA
DEF
DEL
DIM
DRAW

END
EXP

FLASH
FN
FOR
FRE

GET
GOSUB
GOTO
GR

HCOLOR=
HGR
HGR2
HIMEM:
HLIN
HOME
HPLOT
HTAB

IF
IN#
INPUT
INT( )
INVERSE

LEFT$( )
LEN( )
LET
LIST
LOAD
LOG
LOMEM:

MID$( )

NEW
NEXT
NORMAL
NOT
NOTRACE

ON
ONERR
OR

PDL
PEEK
PLOT
POKE
POP
POS
PRINT
PR#

READ
RECALL
REM
RESTORE
RESUME
RETURN
RIGHT$( )
RND( )
ROT=
RUN

SAVE
SCALE=
SCRN( )
SGN
SHLOAD
SIN
SPC( )
SPEED=
SQR
STEP
STOP
STORE
STR$( )

TAB( )
TAN
TEXT
THEN
TO
TRACE

USR

VAL( )
VLIN
VTAB

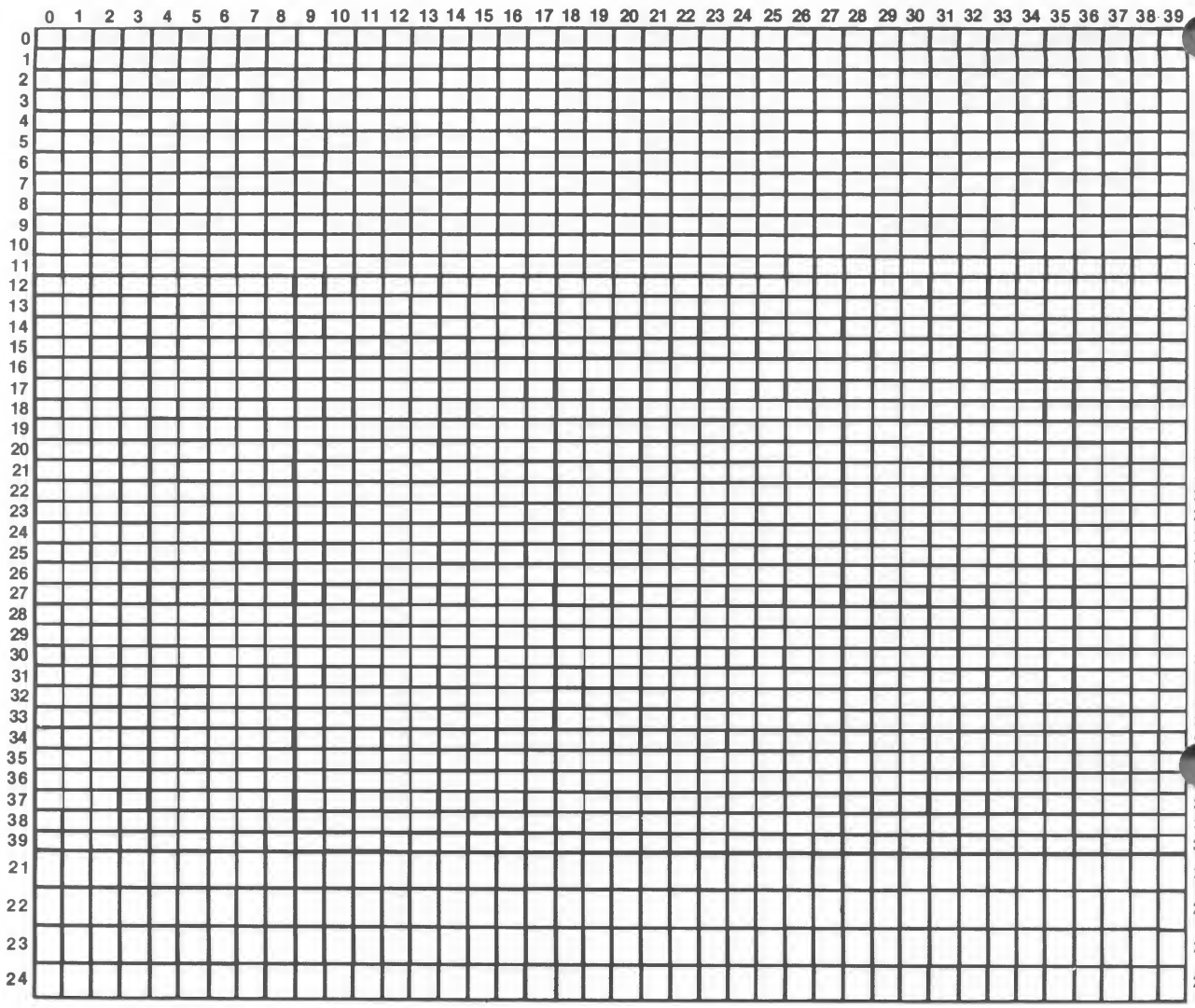
WAIT

XPLOT
XDRAW

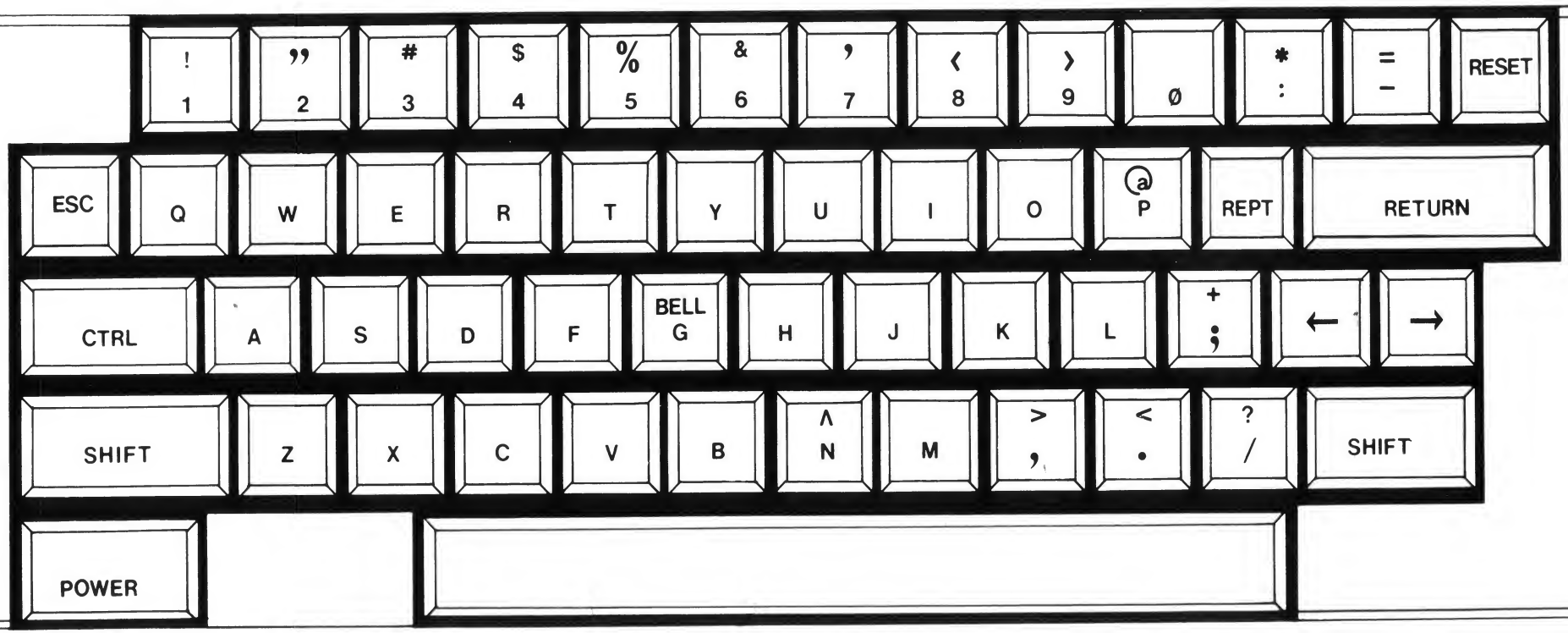
COLOR NUMBER CHART

0 = BLACK
1 = MAGENTA
2 = DARK BLUE
3 = PURPLE
4 = DARK GREEN
5 = GREY
6 = MEDIUM BLUE
7 = LIGHT BLUE
8 = BROWN
9 = ORANGE
10 = GREY
11 = PINK
12 = GREEN
13 = YELLOW
14 = AQUA
15 = WHITE

LO-RESOLUTION SCREEN GRID



KEYBOARD


! 1
" 2
3
$ 4
% 5
& 6
' 7
(8
) 9
Ø
* :
= -
RESET
ESC
Q
W
E
R
T
Y
U
I
O
@ P
REPT
RETURN
CTRL
A
S
D
F
BELL G
H
J
K
L
+ ;
←
→
SHIFT
Z
X
C
V
B
^ N
M
> ,
< .
? /
SHIFT
POWER

Answer To Assignment For Lesson One

Viewing Time 1:15

The assignment for Lesson One was to write a greeting program using the commands learned in Lesson One.

The program was to clear the screen and print the name of the greeting program and the student's name on the screen.

The student was then to use the greeting program to initialize a diskette.

```
10  HOME
20  PRINT "HELLO"
30  PRINT "VIDEOSOFT DISKETTE"
40  PRINT "BEN"
```

The student's program should contain these elements:

Each program line should begin with a line number.

The first line of the program must use the command HOME to clear the screen.

The next three lines should use the PRINT command to display: the name of the greeting program, a line identifying the diskette, and a line that displays the student's name.

To initialize a diskette, the command INIT should have been used followed by the name of the greeting program.

Answer To Assignment For Lesson Two

Viewing Time 3:35

The assignment for Lesson Two was to write a program which printed a short narrative about a hobby or a pet.

The program was to include: one line printed in INVERSE, one line printed in FLASH mode, and one line printed in NORMAL print mode.

The program was to use the SPEED command at least two times; once to slow down the printing speed, and once to return the print speed to normal.

Line 20 was to have been a remark including the SAVE command.

Finally, the student was to SAVE the program on diskette.

```
10  REM  :: HOBBY PROGRAM
20  REM  SAVE FISHING
1000  HOME
1010  PRINT "MY HOBBY IS FISHING.. "
1020  PRINT
1030  PRINT "I LIKE TO FISH ANYTIME."
1040  PRINT "EXCEPT WHEN THERE IS ..."
```

```
1050  PRINT
1060  FLASH
1070  SPEED= 0
1080  PRINT " ** LIGHTNING ** "
1090  SPEED= 255
1100  PRINT
1110  INVERSE
1120  PRINT "FISHING IS GREAT FUN"
1130  NORMAL
1140  END
```

Line 10 is a brief program description in a remark.

Line 20 is the required remark including the SAVE command.

Line 1000 clears the screen with the HOME command.

Line 1010 prints "MY HOBBY IS FISHING."

The PRINT command by itself in line 1020 skips a space between lines.

Line 1030 prints the sentence "I LIKE TO FISH ANYTIME."

Line 1040 prints "EXCEPT WHEN THERE IS..."

Line 1050 prints another blank line.

Line 1060 changes the print mode to FLASHing.

Line 1070 uses the formula "SPEED=0" to decrease the print speed.

Line 1080 prints "LIGHTNING" and the asterisk symbols in flashing mode at zero speed.

Line 1090 returns the computer to normal speed.

Line 1100 prints another blank line.

Line 1110 changes the print mode to INVERSE.

Therefore, line 1120 will PRINT "FISHING IS GREAT FUN" in INVERSE mode at NORMAL speed.

The NORMAL command in line 1130 returns the computer to NORMAL print mode.

Line 1140 ends the program.

Answer To Assignment For Lesson Three

Viewing Time 4:27

The assignment for Lesson Three was to write a data entry program.

In line 500, the program was to assign the string literal "THANK YOU" to a character variable.

Using prompted inputs, the program was to instruct the operator to enter a name, an address and a zip code.

The program was to store the name and address in character variables, and zip code in a numeric variable.

In response to the name entry, the computer was to have printed "THANK YOU" and then PRINT the name.

In response to the address entry, the computer was to have printed "THANK YOU", "YOU LIVE AT", and the address.

In response to the zip code entry, the computer was to have printed "THANK YOU", "YOUR ZIP CODE IS" and the zip code.

The student was to include a program title in a remark at the beginning of the program, and the SAVE command in a remark in line 20.

```
10  REM :: ASSIGNMENT #3
20  REM  SAVE DATA ENTRY
500 A$ = "THANK YOU"
1000  HOME
1010  INPUT "ENTER NAME ";N$
1020  PRINT A$
1030  PRINT N$
1040  INPUT "ENTER ADDRESS ";AD$
1050  PRINT A$
1060  PRINT "YOU LIVE AT "
1070  PRINT AD$
1080  INPUT "ENTER ZIP CODE ";ZIP
1090  PRINT A$
1100  PRINT "YOUR ZIP CODE IS"
1110  PRINT ZIP
1120  END
```

Line 10 is a remark with a brief title for the program.

Line 20 is a remark with the SAVE command and the program file name, "DATA ENTRY".

At line 500 the string literal "THANK YOU" is assigned to the variable A$.

The main portion of the program begins after the remarks and the assignment of the constant.

The HOME comand in line 1000 clears the screen.

The prompted input in line 1010 first instructs the operator to enter a name. The name entered from the keyboard will be assigned to the variable N$.

Line 1020 prints the value of A$, which is "THANK YOU".

Line 1030 prints the value of N$, the name entered by the operator.

Line 1040 uses a prompted INPUT to instruct the operator to enter an address, and then stores the address in the variable AD$.

Line 1050 again prints "THANK YOU".

Line 1060 prints "YOU LIVE AT ".

Line 1070 prints the value of AD$, the address entered by the operator.

Line 1080, uses another prompted INPUT to ask for the zip code and assign it to ZIP. (Zip is used as a numeric variable because it stands for the zip code. The computer will only recognize the first two letters, ZI.)

In line 1090, the program prints "THANK YOU".

Line 1100 prints "YOUR ZIP CODE IS".

Line 1110 prints the value of "ZIP".

The program ends in line 1120.

Answer To Assignment For Lesson Four

Viewing Time 10:33

In Lesson Four the student was given two assignments: the first was a math assignment, the second covered the formatted printing concepts.

First Assignment Viewing Time 4:28

In the math assignment the student was to write a program using the calculating functions.

The program was to instruct the operator to enter a number and perform the following calculations with the number entered:

(A) multiply the number by 5.64.

(B) subtract 6 from the number.

(C) raise to the fourth power.

(D) in a single statement line, divide the number by 7 and add 204.

After each calculation the program was to PRINT a statement describing the calculation and print the answer.

```
10  REM :: ASSIGNMENT # 4
20  REM  SAVE MATH ASSIGNMENT
1000  HOME
1010  INPUT "PLEASE ENTER A NUMBER ";NU
1020  PRINT
1100 M = NU * 5.64
1110  PRINT "THE NUMBER TIMES 5.64 IS"
1120  PRINT M
1200 S = NU - 6
1210  PRINT "THE NUMBER MINUS 6 IS"
1220  PRINT S
1300 E = NU ^ 4
1310  PRINT "THE NUMBER TO THE 4TH POWER IS"
1320  PRINT E
1400 C = NU / 7 + 204
1410  PRINT "THE NUMBER DIVIDED BY 7 PLUS 204 IS"
1420  PRINT C
1500  END
```

Line 1000 clears the screen.

The operator is prompted to enter a number using a prompted input in line 1010. The answer is stored in the numeric variable NU.

Line 1020 is a PRINT line for readability.

The first calculation is in line 1100. NU is multiplied times 5.64 and the answer is stored in M.

Line 1110 prints "THE NUMBER TIMES 5.64 IS".

The answer is printed in line 1120.

The subtraction section begins with the calculation in line 1200 with the difference stored in S.

The print statement in 1210 explains the calculation, "THE NUMBER MINUS 6 IS".

The answer is printed in 1220.

Line 1300 is the exponentiation calculation. NU, the number entered by the operator, is raised to the fourth power, or multiplied by itself four times. The answer is stored in the variable E.

Line 1310 explains the calculation.

Line 1320 prints the answer.

The multiple calculation is performed in line 1400 with the answer stored in the variable C.

To process line 1400, the computer first scans the line from left to right searching for an exponentiation operation. Finding none, the computer again scans the line looking for multiplication or division. Since the first operation is division, the computer divides NU, the number entered by the operator, by seven. As there are no other multiplication or division operations the computer searches for addition or subtraction. Finding an addition calculation, the computer adds 204 to the quotient of the division operation. The answer is stored in the variable C.

Line 1410 explains the calculation.

The answer is printed in line 1420.

The program ends in line 1500.

Second Assignment Viewing Time 6:05

In the formatted printing assignment for Lesson Four, the student was to write a screen formatting program according to a very specific set of instructions.

```
10  REM :: ASSIGNMENT #4-B
20  REM  SAVE NAME ENTRY
1000  HOME
1010  VTAB 3
1020  PRINT  TAB( 10);"NAME ENTRY PROGRAM"
1030  VTAB 5
1040  PRINT "NAME..."
1050  PRINT "AGE ..."
1060  PRINT "PHONE.."
1070  VTAB 10
1080  INPUT "ENTER NAME..";N$
1090  VTAB 5
1100  HTAB 10
1110  PRINT N$
1120  VTAB 11
1130  INPUT "ENTER AGE...";A
1140  VTAB 6
1150  HTAB 10
1160  PRINT A
1170  VTAB 12
1180  INPUT "ENTER PHONE.";P$
1190  VTAB 7
1200  HTAB 10
1210  PRINT P$
1220  VTAB 15
1230  PRINT  TAB( 5);"THANK YOU ";N$
1240 NA = A + 10
```

```
1250  VTAB 19
1260  PRINT "IN 10 YEARS YOU WILL BE ";NA
1270  VTAB 22
1280  PRINT "PROGRAM","ENDED"
1290  END
```

Line 1000 clears the screen and homes the cursor with the HOME command.

Line 1010 positions the cursor to line three, and line 1020 uses the TAB command to print the title at position 10.

The program was to PRINT "NAME", "AGE" and "PHONE" on lines 5, 6, and 7. Since a PRINT command automatically positions the cursor to the next line after printing, only the VTAB 5 command in line 1030 is needed to position the printing of "NAME". "AGE" and "PHONE" will be positioned on the proper lines automatically.

Line 1070 positions the cursor to line 10, so the prompted INPUT in line 1080 will instruct the operator to enter his or her name on line 10.

The VTAB 5, HTAB 10 in lines 1090 and 1100 position the cursor near the top of the screen.

Line 1110 will PRINT the name entered by the operator.

Line 1120 vertically tabs down to line 11 to instruct the operator, in line 1130, to enter his or her age.

The VTAB 6, HTAB 10 in lines 1140 and 1150 place the cursor at line 6, position 10 to PRINT the age in line 1160.

The VTAB 12 in 1170 moves the cursor to line 12 where the operator is instructed to enter his or her phone number with the prompted INPUT in line 1180.

Lines 1190 and 1200 move the cursor to line 7, VTAB 7, and position 10, HTAB 10, which is where the phone number will be printed in line 1210.

Line 1220 positions the cursor at screen line 15 with the VTAB 15 command.

In line 1230, the first TAB command moves the cursor to position 5 by printing blanks. The program then prints "THANK YOU" followed by the name entered by the operator. The semicolons place the variable next to the literal on the same print line.

In line 1240, the program adds 10 to the age entered by the operator and stores the sum in the variable NA.

At line 1250, the program positions the cursor to screen line 19 and line 1260 prints "IN 10 YEARS YOU WILL BE" and the value of the variable NA.

Finally, the program uses the VTAB command in line 1270 to position the cursor to line 22 and line 1280 uses a comma to PRINT "PROGRAM" in print zone one and "ENDED" in print zone two.

Line 1290 ends the program.

Answer To Assignment For Lesson Five

Viewing Time 4:28

The assignment for Lesson Five was to write a multiple choice quiz. The student's program was to include one or more multiple choice questions. For each question there were to be four choices. One correct answer and three incorrect answers. The computer was to respond to each entry. The operator was to answer using a prompted INPUT in each question.

The student was then to use a series of IF-THEN tests to evaluate the operator's answer and make an appropriate comment for each possible answer.

The VideoSoft answer includes two more questions about football.

```
10  REM :: SMALL FOOTBALL QUIZ
20  REM  SAVE DECISION ASSIGNMENT
1000  HOME
1005  INVERSE
1010  PRINT "A FOOTBALL TEAM HAS HOW MANY PLAYERS?"
1020  NORMAL
1025  PRINT
1030  PRINT " A. TWO PLAYERS."
1040  PRINT " B. FIVE PLAYERS."
1050  PRINT " C. NINE PLAYERS."
1060  PRINT " D. ELEVEN PLAYERS."
1070  PRINT
1080  INPUT "ENTER A, B, C OR D: ";AN$
1090  PRINT
1099  REM  :: DECISION TIME ::
1100  IF AN$ = "A" THEN  PRINT "NO! THAT IS A TENNIS TEAM.."
1110  IF AN$ = "B" THEN  PRINT "NO! THAT IS A BASKETBALL TEAM ..."
1120  IF AN$ = "C" THEN  PRINT "NO! THAT IS A BASEBALL TEAM ..."
1130  IF AN$ = "D" THEN  PRINT "YES! THAT IS CORRECT"
1999  REM  :: SECOND QUESTION ::
2000  VTAB 12
2005  INVERSE
2010  PRINT "WHAT SHAPE IS A FOOTBALL?"
2020  NORMAL
2025  PRINT
2030  PRINT "1. SMALL AND ROUND"
2040  PRINT "2. LARGE AND ROUND"
2050  PRINT "3. SMALL AND FLAT"
2060  PRINT "4. OR LARGE AND OBLONG"
2070  PRINT
2080  INPUT "ENTER 1, 2, 3 OR 4: ";NU
2090  PRINT
2099  REM  :: CHECK ROUTINE ::
2100  IF NU = 1 THEN  PRINT "NO! THAT IS A BASEBALL.."
2110  IF NU = 2 THEN  PRINT "NO! THAT IS A BASKETBALL ..."
2120  IF NU = 3 THEN  PRINT "NO! THAT IS A HOCKEY PUCK.."
2130  IF NU = 4 THEN  PRINT "YES! THAT IS CORRECT.."
2200  END
```

In lines 1000 through 1010, the screen is cleared and the first question is printed, "A FOOTBALL TEAM HAS HOW MANY PLAYERS?", in INVERSE.

Line 1020 returns the computer to NORMAL mode.

Line 1025 prints a blank line.

In line 1030 through 1060 the program prints the four possible answers: two, five, nine and eleven.

Line 1070 prints a blank line.

The operator is prompted to answer in line 1080. The answer is stored in the variable AN$.

Line 1090 prints a blank line.

Next we use four IF-THEN commands to test the operator's answer against the four possible answers: "A", "B", "C" and "D". Notice the variable AN$ is compared with the possible answer: the A, B, C and D are placed in quotes. In each case the instruction following the THEN portion of the command prints an appropriate response.

In lines 2000 through 2010 the second question is also printed in INVERSE on screen line 12.

In lines 2020 through 2070 the four possible answers are printed in NORMAL mode and labeled 1, 2, 3 and 4.

The operator is prompted to enter one of the numbers in line 2080, and the answer is stored in the numeric variable NU.

In lines 2100 through 2130 the program tests the operator's answer against each of the four choices. To perform the tests, the program compares the numeric variable NU against the numbers one, two, three and four. Since it is a numeric comparison, the numbers are not placed in quotes.

As in the first question, the THEN portion of the command is followed with an instruction to print an appropriate response.

The program ends in line 2200.

Answer To Assignment For Lesson Six

Viewing Time 11:45

The assignment for Lesson Six was to write a long program: "TWENTY-THREE MATCHES".

The program was divided into six sections.

```
10  REM :: MATCHES GAME
20  REM  SAVE MATCHES
40  REM :: HOUSEKEEPING
100 BL$ =  CHR$ (7)
990  REM :: TITLE PAGE
1000  HOME : VTAB 4: FLASH
1010  PRINT "!!! MATCHES !!!"
1020  NORMAL : VTAB 8: PRINT "A GAME FOR TWO PLAYERS"
1100  VTAB 12
1110  INPUT "PLAYER #1, ENTER NAME: ";P1$
1120  INPUT "PLAYER #2, ENTER NAME: ";P2$
1200 ML = 23
1990  REM :: INSTRUCTIONS
2000  HOME : VTAB 4: INVERSE
2010  PRINT " INSTRUCTIONS ": NORMAL
2020  VTAB 8
2030  PRINT "IN THIS GAME THERE ARE"
2040  PRINT "23 MATCHES ...": PRINT
2050  PRINT "EACH PLAYER MAY TAKE"
2060  PRINT "AS FEW AS 1 MATCH..."
2070  PRINT "OR AS MANY AS 3 MATCHES."
2080  PRINT : PRINT "THE PLAYER TAKING THE"
2090  PRINT "LAST MATCH LOSES."
2100  VTAB 20: INPUT "PRESS RETURN TO BEGIN: ";A$
2990  REM :: PLAYER 1'S TURN
3000  HOME : INVERSE : VTAB 4
3010  PRINT ML;" MATCHES LEFT."
3100  NORMAL : VTAB 8
3110  PRINT P1$;" TAKE SOME MATCHES"
3120  INPUT "1, 2 OR 3 MATCHES: ";MT
3130  IF MT = 1 OR MT = 2 THEN 3200
3140  IF MT = 3 THEN 3200
3150  GOTO 3120
3200 ML = ML - MT
3210  IF ML > 0 THEN 4000
3220  HOME : PRINT BL$: VTAB 10: HTAB 10
3230  FLASH : PRINT P1$;" LOSES": NORMAL
3240  GOTO 9000
3990  REM :: PLAYER 2'S TURN
4000  HOME : INVERSE : VTAB 4
4010  PRINT ML;" MATCHES LEFT."
4100  NORMAL : VTAB 8
4110  PRINT P2$;" TAKE SOME MATCHES"
4120  INPUT "1, 2 OR 3 MATCHES: ";MT
4130  IF MT = 1 OR MT = 2 THEN 4200
4140  IF MT = 3 THEN 4200
4150  GOTO 4120
4200 ML = ML - MT
4210  IF ML > 0 THEN 3000
```

```
4220  HOME : PRINT BL$: VTAB 10: HTAB 10
4230  FLASH : PRINT P2$;" LOSES": NORMAL
8990  REM :: END OF PGM
9000  VTAB 18: PRINT "DO YOU WANT TO PLAY AGAIN?"
9010  VTAB 20: INPUT "ENTER Y OR N: ";AN$
9020  IF AN$ = "Y" THEN 1000
9030  IF AN$ <  > "N" THEN 9010
9040  HOME : PRINT BL$: VTAB 12: HTAB 10
9050  PRINT "GOODBYE"
9060  END
```

SECTION ONE: HOUSEKEEPING (Lines 10-100)

Lines 10 and 20 are the normal opening remarks.

The remark in line 40 identifies this as the housekeeping section.

In line 100, the bell character, CHR$(7), is assigned to the variable BL$.

SECTION TWO: INTRODUCTION (Lines 990-1200)

The remark in line 990 documents the section: TITLE PAGE

Line 1000 is a multiple statement line which clears the screen, VTABs to line 4 and switches to FLASHing mode.

Line 1010 prints the game title, "!!! MATCHES !!!".

Line 1020 switches back to NORMAL, VTABs to line 8 and prints "A GAME FOR TWO PLAYERS".

After VTABing to line 12, line 1110 uses a prompted INPUT to accept the name of player one which is stored in P1$.

The prompted INPUT in line 1120 accepts the name of player two and stores it in P2$.

In line 1200 the starting value of the number of matches is established by assigning 23 to variable ML.

SECTION THREE: INSTRUCTIONS (Lines 1990-2100)

Line 2000 clears the screen, VTABs to line 4 and switches to INVERSE mode.

Line 2010 prints "INSTRUCTIONS" and switches back to NORMAL mode.

Line 2020 VTABs to line 8.

Lines 2030 through 2090 print the instructions for the game.

Notice, on lines 2040 and 2080, an extra PRINT statement is included for readability.

Line 2100 VTABs to line 20 and uses a dummy INPUT to hold the instructions on the screen.

SECTION FOUR: PLAYER ONE'S TURN (Lines 2990-3240)

Line 3000 clears the screen, switches to INVERSE mode and VTABs to line 4.

Line 3010 prints the value of ML and the phrase "MATCHES LEFT". The variable ML is used throughout to keep track of the number of matches remaining.

Line 3100 switches back to NORMAL and VTABs to screen line 8.

Line 3110 prints the name of player one, P1$, followed by the instructions: "TAKE SOME MATCHES".

In line 3120 we prompt player one to take 1, 2 or 3 matches. The number of matches taken is stored in the variable MT.

Line 3130 uses an IF-THEN with an OR to test if MT is equal to one or two.

Line 3140 tests to see if MT is equal to three.

Both IF-THEN commands send the program to line 3200 when the conditions set in the IF portion of the command are met. So, if MT, matches taken, is equal to one, two or three, the computer processes line 3200. If not, line 3150 will be executed sending the program back to line 3120 which instructs player one to enter one, two or three matches.

Line 3200 instructs the computer to subtract the MT, matches taken, from ML, matches left.

Line 3210 tests to see if there are any matches left; is ML greater than zero? If any matches remain, the program executes line 4000, the beginning of player two's turn.

If no matches remain, line 3220 is executed. This multiple statement line clears the screen and sounds the bell by printing BL$. Recall, in the beginning of the program we assigned the bell character, CHR$(7), to BL$.

Line 3220 then VTABs to line 10 and HTABs to position 10.

Line 3230 switches to FLASHing, prints player one loses and switches back to NORMAL.

Line 3240 sends the program to line 9000, the ending section.

SECTION FIVE: PLAYER TWO'S TURN (Lines 3990-4230)

Player two's turn is a duplicate of player one's turn with four differences.

First, while the commands are the same, the line numbers are different.

Second, we print player two's name, P2$ instead of P1$.

Third, in line 4210, if any matches are left the program goes to line 3000, the beginning of player one's turn.

The fourth and final difference is that player two's turn is one line shorter than player one's turn. If player one lost, we instructed the computer to by-pass player two's turn and go to the ending section. If player two loses, the ending section is next so we do not need the GOTO.

SECTION SIX: ENDING SECTION (Lines 8990-9060)

In line 9000 we VTAB to line 18 and PRINT "DO YOU WANT TO PLAY AGAIN?"

Line 9010 VTABs to screen line 20 and prompts the players to enter "Y" or "N".

In line 9020 if "Y" is entered, we go back to line 1000 and start the program again.

In line 9030 if an "N" was not entered, the program returns to line 9010 and again prompts the players to enter "Y" or "N".

Line 9040 will only be executed if the players do not want to play again. The line clears the screen, sounds the bell, VTABs to line 12 and HTABs to position 10.

In line 9050, we PRINT "GOODBYE", and END the program in line 9060.

Answer To Assignment For Lesson Seven

Viewing Time 10:27

In Lesson Seven on FOR-NEXT loops, the student was given two short assignments.

In the first assignment the student was to write a self-destruct program. In the second, the student was to write a program to PRINT asterisks in every other position, on every other line of the CRT.

Assignment One Viewing Time 4:41

```
10  REM :: SELF DESTRUCT PROGRAM
20  REM  SAVE DESTRUCT
100 B$ = "BOOM"
1000  FOR I = 10 TO 0 STEP  - 1
1010  HOME
1020  VTAB 12: HTAB 7
1030  PRINT I;" SECONDS TO SELF-DESTRUCT "; CHR$ (7)
1040  FOR J = 1 TO 740
1050  NEXT J
1060  NEXT I
2000  HOME : FLASH
2010  FOR I = 23 TO 1 STEP  - 1
2020  VTAB I: PRINT B$,B$,B$
2030  NEXT I
2040  VTAB 24
2050  NORMAL
2060  END
```

In line 100, the word "BOOM" is assigned to the variable B$. Later, B$ will be used to PRINT the word "BOOM".

Lines 1000 through 1060 are the count-down section of the program.

The first loop in our program, the "I" loop, has the FOR command on line 1000 which sets a beginning value of ten, and an ending value of zero. STEP minus one instructs the computer to subtract one from the value of "I" each time the loop repeats. The second command of the pair, NEXT I, is located on line 1060. This "I" loop simply instructs the computer to count down from ten to zero.

This meets the requirement that the program use a loop with a negative STEP to count down the seconds.

Line 1010 clears the screen.

Line 1020 places the cursor on line twelve position seven.

Line 1030 instructs the computer to PRINT the value of "I" on the same line with the message "SECONDS TO SELF-DESTRUCT", and "CHR$(7)" which makes the computer beep each time the loop repeats.

To hold the message on the screen for one second, a timing loop is used in lines 1040 and 1050.

The NEXT I command in line 1060 marks the end of the count-down section of the program.

Lines 2000 through 2030 are the "BOOM" section of the program.

Line 2000 clears the screen and, on the same line, sets the computer to FLASH mode.

To PRINT the "BOOMS" moving from the bottom of the screen to the top of the screen, line 2010 establishes a FOR-NEXT loop with a negative STEP. The loop will decrease the value of "I" from 23 down to 1.

In line 2020, the VTAB command is used to position the cursor to the screen line which corresponds with the value of "I", and then PRINTs B$ three times using a comma to separate the variables. Using the commas, the three "BOOMS" will PRINT in the three screen print zones.

The NEXT I command in line 2030 subtracts one from the value of I and instructs the computer to process the next pass of the loop.

The VTAB 24 in line 2040 merely positions the cursor to the bottom of the screen before returning the computer to NORMAL print mode and ending the program in lines 2050 and 2060.

Second Assignment Viewing Time 5:46

The second assignment was designed to let the student use a nested loop to control the position of the cursor on the CRT.

The student was to use a nested loop in which the outside loop controlled the vertical position of the cursor on the screen line. The inside loop was to control the horizontal position of the cursor.

The student was to PRINT asterisks in every other position on every other line of the CRT. To accomplish this, a student could use either the STEP command or could calculate the VTAB and HTAB positions.

The course answer uses both techniques.

FIRST SOLUTION: LINES 1000-1070

This is the solution to the assignment using the STEP command.

```
10   REM :: ASSIGNMENT #7B
20   REM  SAVE ASTERISK 1
1000   HOME
1010   FOR I = 1 TO 23 STEP 2
1020   VTAB I
1030   FOR J = 1 TO 39 STEP 2
1040   HTAB J
1050   PRINT "*";
1060   NEXT J
1070   NEXT I
9000   END
```

After clearing the screen, the outside loop begins at one and ends at 23, STEPping by twos.

In line 1020 the command VTAB I positions the cursor to every other line since the STEP 2 command instructs the computer to increase the value of I by two for each pass. I will equal one, then three, and then five, and the other odd numbers up to 23.

While the cursor is positioned on each line, the inside loop will complete its entire cycle.

Line 1030 establishes the inside loop with a beginning value of one and ending value of 39, again STEPing by twos.

On line 1040, we HTAB to position J which will take on each of the odd number values from one to 39. The value of J will first move the cursor to position one, then three and so on until the value of J is 39.

In line 1050, we PRINT "*". The semicolon instructs the computer to place the asterisks on the same line.

The NEXT J command in line 1060 adds two to the value of J and instructs the computer to proceed with the next pass of the inside loop.

The NEXT I command in line 1070 adds two to the value of I and instructs the computer to process the next pass of the outside loop.

Line 9000 ends the program.

SECOND SOLUTION: LINES 2000-2060

The second solution uses a calculation to position the cursor.

```
10  REM :: ASSIGNMENT #7B
20  REM  SAVE ASTERISK 2
2000  HOME
2010  FOR I = 1 TO 11
2020  VTAB I * 2
2030  FOR J = 1 TO 19
2040  HTAB J * 2: PRINT "*";
2050  NEXT J
2060  NEXT I
9000  END
```

In this solution, the outside loop is established with a beginning value of 1 and an ending value of 11. To position the cursor to every other line, the computer multiplies the value of I, the loop variable, by two in program line 2020. Then the VTAB command is used to position the cursor.

Thus, on the first pass, I will equal one and the cursor will be positioned on line "one times two", line two. On the second pass, I will equal two. So, the calculation will be "two times two", line four. The outside loop will position the cursor on each of the even numbered screen lines.

The inside loop is established in line 2030 with a beginning value of 1 and an ending value of 19.

The HTAB command in line 2040 multiplies the value of J times two when calculating the horizontal position of the cursor. Thus, on each even screen line, the inside J-loop will position the cursor at each of the even positions.

Line 2040 PRINTs an asterisk, again using the semicolon to instruct the computer to put the asterisks on the same line.

Lines 2050 and 2060 are the required NEXT commands. First the NEXT command for the inside loop, then the NEXT command for the outside loop.

Line 9000 ends the program.

Answer to Assignment For Lesson Eight

Viewing Time 6:18

In Lesson Eight the student was asked to write a program which acted as a four function calculator.

The student was to first accept two numbers from the operator and END the program if both the numbers entered were zero. If one or both of the numbers was other than zero, the program was to allow the operator to choose the operation to perform.

Upon the operator's selection, the program was to clear the screen and use a computed GOSUB to perform either the add, subtract, multiply or divide routine. Each of the four routines was to perform the selected operations with the two numbers entered by the operator, then PRINT the equation and result in the form: "1 + 2 = 3".

After the equation was displayed the program was to use a dummy INPUT to allow the operator to see the answer, and then return to the beginning of the program to allow further entries.

```
10  REM :: CALCULATOR PROGRAM
20  REM   SAVE CALCULATOR
1000  HOME
1010  VTAB 8
1020  PRINT "ENTER TWO NUMBERS FOR THE COMPUTER"
1030  PRINT "TO ADD/SUBTRACT/MULTIPLY/DIVIDE"
1040  VTAB 11
1050  PRINT "ENTER ZEROS TO END"
1060  VTAB 15
1070  INPUT "ENTER FIRST NUMBER ";N1
1080  VTAB 17
1090  INPUT "ENTER SECOND NUMBER ";N2
1100  IF N1 = 0 AND N2 = 0 THEN 9000
1110  HOME
1120  VTAB 8
1130  PRINT "SELECT OPERATION TO PERFORM"
1140  PRINT "  1 - ADD"
1150  PRINT "  2 - SUBTRACT"
1160  PRINT "  3 - MULTIPLY"
1170  PRINT "  4 - DIVIDE"
1180  PRINT
1190  INPUT "ENTER NUMBER OF CHOICE ";C
1200  IF C < 1 OR C > 4 THEN 1110
1210  HOME
1220  ON C GOSUB 2010,3010,4010,5010
1230  VTAB 20
1240  INPUT "PRESS RETURN TO CONTINUE ";N$
1250  GOTO 1000
2000  REM : ADD ROUTINE
2010 R = N1 + N2
2020  VTAB 10
2030  PRINT N1;" + ";N2;" = ";R
2040  RETURN
3000  REM : SUBTRACT ROUTINE
3010 R = N1 - N2
3020  VTAB 10
3030  PRINT N1;" - ";N2;" = ";R
3040  RETURN
4000  REM : MULTIPLY ROUTINE
4010 R = N1 * N2
```

```
4020  VTAB 10
4030  PRINT N1;" * ";N2;" = ";R
4040  RETURN
5000  REM : DIVIDE ROUTINE
5010 R = N1 / N2
5020  VTAB 10
5030  PRINT N1;" / ";N2;" = ";R
5040  RETURN
9000  HOME
9010  PRINT "THE PROGRAM HAS ENDED"
9020  END
```

Lines 10 and 20 are the opening remarks.

Line 1000 and 1010 clear the screen and position the cursor on line eight.

The instructions to the operator are printed by lines 1020 through 1050.

Line 1020 prints "ENTER TWO NUMBERS FOR THE COMPUTER".

Line 1030 prints "TO ADD / SUBTRACT / MULTIPLY / DIVIDE".

Line 1040 repositions the cursor on screen line eleven.

Line 1050 prints "ENTER ZEROS TO END".

Line 1060 positions the cursor on screen line 15.

Line 1070 prompts the operator on screen line 15 for the first number and places it in the variable N1.

Line 1080 positions the cursor on screen line 17.

Line 1090 instructs the operator to enter the second number and stores it in N2.

If both of the numbers entered are zero, line 1100 instructs the computer to jump to line 9000 for the ending section of the program.

Line 1110 clears the screen.

Line 1120 positions the cursor on screen line 8.

The PRINT statements in lines 1130 through 1170 instruct the operator to make a selection between operations. This programming technique is called a "MENU".

Line 1180 prints a blank line.

The prompted INPUT in line 1190 instructs the operator to enter the number of the selection and stores it in the variable C.

In line 1200 we check the entry. If the number entered is less than one or greater than four the computer will return to 1110.

Line 1210 clears the screen to prepare for the equation PRINT.

Line 1220 executes the subroutine which performs the selected operation using a computed GOSUB.

For example, if the operator enters "1", the computer will execute the first subroutine on line number 2010, and so on.

When the computer returns from the subroutine in line 1230, we use a VTAB command to position the cursor at line 20.

In line 1240, we use a dummy INPUT to hold the calculation result on the screen until the operator presses return.

Line 1250 instructs the computer to return to line 1000 so the operator can enter more numbers for calculations.

Line 2000 labels the add routine.

Line 2010 adds the two numbers, N1 and N2, and then stores the result in the variable R.

Line 2020 VTABs to line 10.

Line 2030 prints the first number, a space for readability, the plus sign, another space, the second number, an equal sign and finally the result of the equation.

Line 2040 is the RETURN command which ends the addition subroutine.

The subtraction routine is in lines 3000-3040 and follows the same order as the add routine.

First the numbers are subtracted. Then the calculation is printed on line 10, and the RETURN command ends the subroutine in line 3040.

Lines 4000-5040 contain the multiplication routine and the division routine. Each follows the same format as the other two subroutines: first performing the calculation, and then printing the equation on screen line 10.

The ending section of the program begins in line 9000.

The ending section clears the screen, prints the message "THE PROGRAM HAS ENDED" and ENDs the program.

Answer To Assignment For Lesson Nine

Viewing Time 7:15

In Lesson Nine the student was to write a program which allowed the operator to enter the won, lost and tied records for four baseball teams.

The student was to place the team names in a DATA statement and READ them into an array using a loop.

The student was to write an input section in which the operator enters the records for each team.

The team records were to be stored in a multi-dimensional array.

Then a print section was to use a nested loop to print the records.

```
10  REM :: ARRAY ASSIGNMENT
20  REM  SAVE TEAM RECORDS
50  DIM T$(3),T(3,2)
100  DATA  YANKEES,DODGERS,RANGERS,CUBS
110  FOR I = 0 TO 3
120  READ T$(I)
130  NEXT I
1000  HOME : PRINT "ENTER TEAM RECORDS": VTAB 4: INVERSE
1010  PRINT "TEAM"; TAB( 20);"WON   LOST  TIED"
1020  NORMAL
1030  FOR I = 0 TO 3
1040  VTAB I + 5: PRINT T$(I);
1050  FOR J = 0 TO 2
1060  VTAB I + 5: HTAB J * 6 + 20
1070  INPUT T(I,J)
1080  NEXT J
1090  NEXT I
1890  REM :: PRINT SCREEN TITLE
1900  HOME : VTAB 5: HTAB 10
1910  PRINT " TEAM RECORDS "
1920  VTAB 8: INVERSE
1930  PRINT "TEAM"; TAB( 20);"WON   LOST  TIED"
1940  NORMAL
2000  REM  :: PRINT RECORDS ::
2010  FOR I = 0 TO 3
2020  VTAB I + 10: HTAB 1
2030  PRINT T$(I);
2040  FOR J = 0 TO 2
2050  HTAB J * 6 + 20
2060  PRINT T(I,J);
2070  NEXT J
2080  NEXT I
2090  END
```

Lines 10 and 20 are the standard opening remarks.

In line 50 of the housekeeping section, we dimension the two arrays. The character array T$ has four elements. The numeric array T has four rows with three elements each.

The names of the teams are stored in the DATA line 100, and READ into the T$ array through the loop in lines 110, 120 and 130.

Lines 1000-1090 are the input section.

Line 1000 is a multiple statement line which clears the screen, prints "ENTER TEAM RECORDS", VTABs to line 4 and switches to INVERSE mode.

In line 1010, we PRINT the column headings.

Line 1020 returns the computer to NORMAL.

The I-loop in line 1030 controls three items: the printing of the team names, the calculation of the screen lines for VTAB, and the selection of the row of the multi-dimensional array.

Line 1040 VTABs to row "I+5" which, on the first pass, will be row five and then prints the name of the team.

The J-loop begins in line 1050. The J-loop controls two operations: the calculation of the HTAB position, and the selection of which element is entered by the operator for each row of the multi-dimensional array.

Line 1060 positions the cursor for the operator to INPUT the record in the proper column. The VTAB I+5 keeps the cursor on the same row as the team name. The HTAB J*6+20 positions the cursor to the proper column. The columns start at position 20 and are positioned six spaces apart. As J increases in value by one, the cursor will be positioned six spaces further.

Line 1070 is the INPUT for the operator. While I is equal to the number of each team, the value of J will be 0, 1 and 2 for the array elements corresponding to games won, games lost and games tied.

Lines 1080 and 1090 are the two NEXT commands for the nested INPUT loop.

Lines 1900 through 1940 PRINT the headings at the top of the screen. The program prints "TEAM RECORDS" at position 10 on line 5. On line 8, the program prints the column headings in INVERSE.

The team records are printed in the nested loop on lines 2010 through 2080.

The logic of the printing section is almost identical to the INPUT section.

The program uses the I-loop beginning in line 2010 to control the screen line selection on line 2020 and the printing of the team name on line 2030.

The J-loop beginning on line 2040 controls the HTAB position of the cursor on line 2050.

In line 2060 the program prints the team records. While "I" will control which row of the T-array the program is using, "J" will equal 0, 1 and 2, controlling the elements of each row for games won, lost and tied.

Lines 2070 and 2080 are the NEXT commands for the nested loop.

The program ends in line 2090.

First we extract the letters of the name starting from the second character with the MID$ command. Then we add the first letter to the end with the LEFT$ command.

Lines 3050 and 3060 PRINT this version of the name and ask the operator if it is correct.

Next, we mix up the name by placing the first half of it last.

Line 3070 calculates half the length of the name by dividing the length of the name by two.

Line 3080 concatenates the name using the RIGHT$ and LEFT$ commands.

Lines 3090 and 3100 PRINT this version of the name and ask the operator if it is correct.

Line 3110 prints "I'M SORRY" and the name correctly.

Then in lines 3120 and 3130 we PRINT "I ALWAYS HAVE TROUBLE REMEMBERING NAMES" and "GOODBYE".

Then we PRINT "I ALWAYS HAVE TROUBLE REMEMBERING NAMES" and "GOODBYE".

Line 9000 ends the program.

Answer To Assignment For Lesson Eleven

Viewing Time 1:57

The assignment for Lesson Eleven was to draw a Lo Res picture.

The VideoSoft answer is a picture of a FIRE BIRD.

To draw the fire bird, we deleted the lines which drew the dragon from the dragon picture and replaced them with different HLIN, VLIN and PLOT commands to draw the fire bird.

The technique of using a similar working program as the base for another program is frequently used by professional programmers to improve both their programming speed and the reliability of their programs.

```
10  REM :: FIRE BIRD PICTURE
20  REM  SAVE FIRE BIRD
500  HOME
510  GR
890  REM :: BORDER ::
900  COLOR= 12
910  HLIN 0,39 AT 0
920  HLIN 0,39 AT 38
930  VLIN 0,38 AT 0
940  VLIN 0,38 AT 39
950  COLOR= 13
960  FOR I = 1 TO 37
970  HLIN 1,38 AT I
980  NEXT I
990  REM :: FIRE BIRD ::
1000  COLOR= 2
1010  HLIN 9,13 AT 37: HLIN 26,30 AT 37
1020  HLIN 10,12 AT 36: HLIN 27,29 AT 36
1030  HLIN 11,13 AT 35: HLIN 26,28 AT 35
1040  HLIN 12,17 AT 34: HLIN 22,27 AT 34
1050  HLIN 13,18 AT 33: HLIN 21,26 AT 33
1060  HLIN 14,25 AT 32: HLIN 15,18 AT 31
1070  HLIN 21,24 AT 31: HLIN 15,17 AT 30
1080  HLIN 15,16 AT 29: HLIN 23,24 AT 29
1090  HLIN 2,5 AT 9: HLIN 16,23 AT 9
1100  HLIN 17,22 AT 10: HLIN 35,37 AT 10
1110  HLIN 2,3 AT 11: HLIN 18,21 AT 11
1120  HLIN 36,37 AT 11: HLIN 19,20 AT 12
1130  HLIN 12,13 AT 15: HLIN 26,27 AT 15
1140  HLIN 13,14 AT 16: HLIN 25,26 AT 16
1150  HLIN 14,15 AT 17: HLIN 24,25 AT 17
1160  HLIN 15,16 AT 18: HLIN 23,24 AT 18
1170  HLIN 16,17 AT 19: HLIN 22,23 AT 19
1180  HLIN 10,11 AT 25: HLIN 28,29 AT 25
1190  HLIN 9,10 AT 26: HLIN 29,30 AT 26
1200  HLIN 8,9 AT 27: HLIN 30,31 AT 27
1210  HLIN 7,8 AT 28: HLIN 31,32 AT 28
1220  HLIN 6,7 AT 29: HLIN 32,33 AT 29
1230  HLIN 5,6 AT 30: HLIN 33,34 AT 30
1240  HLIN 4,5 AT 31: HLIN 34,35 AT 31
1250  HLIN 3,4 AT 32: HLIN 35,36 AT 32
1260  HLIN 2,3 AT 33: HLIN 36,37 AT 33
```

```
8120  NEXT I
8130  GOTO 8000
8140  TEXT
8150  HOME
8160  END
```

Lines 10 and 20 are the standard opening remarks.

Line 500 clears the text screen.

Line 510 switches the computer to Lo Res graphics mode.

Lines 900-940 draw a green border using HLIN and VLIN commands.

Lines 950-980 use the COLOR= and HLIN commands in a loop to fill the picture area with yellow, the dominant color of the fire bird.

The actual drawing of the fire bird begins in line 1000.

Lines 1000-1310 use the COLOR=, HLIN and PLOT commands to draw the dark blue portions of the fire bird.

Lines 1400-1460 use the COLOR=, HLIN, VLIN and PLOT commands to draw the orange legs and beak of the bird.

Lines 1500-1780 use the COLOR=, HLIN, VLIN and PLOT commands to carve the outline of the bird in black.

Lines 2000-2290 draw the magenta portions of the fire bird.

Lines 3000-3190 draw the light purple portions of the fire bird.

Lines 4000-4020 draw the white of the bird's eye.

Lines 4100-4110 draw the magenta pupil of the eye.

Lines 4200-4210 add some dark blue to the bird.

Lines 7000-7080 PRINT the picture title, first flashing, then in INVERSE, and finally in NORMAL print mode.

Lines 8000-8130 are the color changing portion of the program and are identical to the color changing module in the dragon picture.

Lines 8140-8160 end the program by switching back to TEXT mode and clearing the screen.

Answer To Assignment For Lesson Twelve

Viewing Time 11:55

In the assignment for Lesson Twelve the student was to use the animation techniques to add a moving block of color to a Lo Res picture.

The student was also to make the computer buzz using the PEEK sound command, and to use the keyboard PEEK to allow the operator to change either the color of the animated object or some other aspect of the picture by pressing a key on the keyboard.

The VideoSoft answer to the assignment draws a star field as a Lo Res picture and moves a white block as a space ship.

```
10  REM :: SPACE SHIP PGM
20  REM  SAVE SPACE SHIP
40  REM :: HOUSEKEEPING SECTION
100 C = 15
120 IK = 201:MK = 205:JK = 202:KK = 203:SB = 160
130 KB =  - 16384:KS =  - 16368
140 SD =  - 16336
150  HOME : GR
190  REM :: MAKE STAR FIELD
200  FOR I = 1 TO 40
210 X =  INT (38 *  RND (1) + 1)
220 Y =  INT (38 *  RND (1) + 1)
230 SC =  INT (14 *  RND (1) + 1)
240  COLOR= SC
250  PLOT X,Y
260  NEXT I
490  REM :: STARTING POSITION
500 X = 20:Y = 20
510 XP = X:YP = Y
520 B =  SCRN( 20,20)
990  REM :: BEGIN ANIMATION
1000 D = 0
1010 PK =  PEEK (KB)
1020  IF PK = IK THEN D = 1: POKE KS,0: GOSUB 3000
1030  IF PK = JK THEN D = 2: POKE KS,0: GOSUB 3100
1040  IF PK = MK THEN D = 3: POKE KS,0: GOSUB 3200
1050  IF PK = KK THEN D = 4: POKE KS,0: GOSUB 3300
1060  IF PK = SB THEN 9000
1070  IF D = 0 THEN 1000
1080  ON D GOTO 1100,1200,1300,1400
1090  REM :: DIRECTION 1=UP
1100 Y = Y - 1: IF Y < 0 THEN Y = 39
1110  GOTO 1500
1190  REM :: DIRECTION 2=LEFT
1200 X = X - 1: IF X < 0 THEN X = 39
1210  GOTO 1500
1290  REM :: DIRECTION 3=DOWN
1300 Y = Y + 1: IF Y > 39 THEN Y = 0
1310  GOTO 1500
1390  REM :: DIRECTION 4=RIGHT
1400 X = X + 1: IF X > 39 THEN X = 0
1500 TB =  SCRN( X,Y)
1510  COLOR= C: PLOT X,Y
1520  COLOR= B: PLOT XP,YP
```

```
1530 XP = X:YP = Y
1540 B = TB
2000  GOTO 1000
2990  REM :: SOUND SUBROUTINE
3000  FOR I = 1 TO 10
3010 S =  PEEK (SD):S =  PEEK (SD)
3020  NEXT I
3030  RETURN
3100  FOR I = 1 TO 10
3110 S =  PEEK (SD)
3120  NEXT I
3130  RETURN
3200 S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):
      S =  PEEK (SD):S =  PEEK (SD)
3210 S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):
      S =  PEEK (SD):S =  PEEK (SD)
3230  RETURN
3300  FOR I = 1 TO 5
3310 S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):
      S =  PEEK (SD):S =  PEEK (SD)
3320  NEXT I
3330  RETURN
9000  VTAB 21: INPUT "PRESS RETURN ";A$
9010  TEXT : HOME
9020  END
```

Lines 10-520 are the housekeeping section of the program.

Lines 10, 20 and 40 are the opening remarks.

Line 100, assigns the color white, 15, to C.

The I, J, K and M keys are used to allow the operator to control the movement of the ship, and the space bar to end the animation.

Therefore, line 120 assigns the numeric codes to the five keys.

"IK", the "I" key is 201.

"MK", the "M" key is 205.

"JK", the "J" key is 202.

"KK", the "K" key is 203.

"SB", the space bar is 160.

In Line 130, the keyboard PEEK and POKE addresses are assigned to KB and KS.

Line 140 assigns the sounds PEEK address to SD.

Line 150 clears the screen and switches to graphics mode.

The star field is drawn by plotting 40 random blocks of color on the CRT.

The loop in line 200 simply controls the number of stars to be plotted, 40 stars.

Line 210 uses the RND( ) and the INT( ) commands to pick a random number between one and 38.

The random number is assigned to X to be used as the horizontal, or column coordinate for each star.

Line 220 uses an identical random function to select a vertical, or row coordinate for each star.

Line 230 uses another random function to assign a color between 1 and 14 to the variable SC. Omit the colors black (zero), and white (15), because black is the dominant background color and white is the color of the space ship.

Line 240 sets the color to SC.

Line 250 plots a random star.

The NEXT I command in line 260 executes the next pass of the loop.

Line 500 sets the starting coordinates at column 20, row 20.

Line 510 assigns the current coordinates to the previous coordinates.

Line 520 stores the first background color using the SCRN( ) command with the coordinates 20,20.

The main program module begins in line 1000.

Lines 1000-1070 allow the operator to set the direction of movement based on the last key pressed by the operator.

Line 1000 initializes the value of D as zero at the beginning of each animation sequence.

Line 1010 PEEKs at the keyboard.

Lines 1020-1060 test the number from the keyboard to see if one of the direction keys was pressed or if the space bar was pressed, and execute a subroutine which creates a slightly different sound for each key.

In line 1020 if the I key was pressed, D is given a value of one (up) and the keyboard is reset with "POKE KS, 0". Then the sound subroutine at 3000 is executed.

Line 1030 assigns a value of 2, left, to the variable D if the J was pressed. Then line 1030 resets the keyboard and executes the sound subroutine in line 3100.

Line 1040 assigns a value of 3, down, to D if the M key was pressed, then resets the keyboard and executes the sound subroutine in line 3200.

Line 1050 assigns a value of 4, right, to D if the K key was pressed, resets the keyboard and executes the sound subroutine in line 3300.

In line 1060, if the space bar was pressed, the program goes to the ending section in line 9000.

The IF-THEN in line 1070 instructs the computer to GOTO line 1000 if no direction has been assigned (D will equal zero).

Lines 1080-1400 were taken verbatim from the "ROAMING BLOCK" program explained in Lesson Twelve.

This section is the direction control section which moves the white block by adding or subtracting from the "X" or "Y" coordinates.

Lines 1500-2000 create the illusion of animation by PLOTing the white block and "erasing" by redrawing in the previous background color.

Lines 1500 uses SCRN( ) to find and temporarily store the background color at coordinates "X, Y".

Line 1510 sets the color to C, 15, and PLOTs the white block at X, Y. (C was assigned the number 15 in line 100.)

Line 1520 sets the color to B, the previous coordinates' background color, and restores the background by PLOTting at XP,YP.

Line 1530 stores the new coordinates, X,Y, in the previous coordinate variables, XP,YP.

Line 1540 stores the temporary background color (TB) in the variable B so it can be restored when the block is moved.

Line 2000 sends the computer to the beginning of the animation sequence for the next movement.

Lines 3000 through 3030 comprise the first sound subroutine executed when the ship moves up.

Line 3000 establishes a FOR-NEXT loop with 10 passes.

Line 3010 executes two sound PEEK commands on each loop pass.

Line 3020 completes the loop with the NEXT I command.

Line 3030 ends the subroutine with the command word RETURN.

Lines 3100-3130 are the second sound routine executed when the ship moves to the left.

These lines are the same as the first subroutine except in line 3110 in which only one sound routine is executed on each of the ten loop passes.

Lines 3200-3230 are the third sound routine which executes if the direction is down.

Line 3200 is a multiple statement line which executes the sound PEEK command five times.

Line 3210 is identical to line 3200.

The subroutine ends with the RETURN command in line 3230.

Lines 3300 through 3320 are the fourth sound subroutine which executes when the ship moves to the right.

Line 3300 establishes a loop with 5 passes.

Line 3310 executes five sound PEEK commands.

The loop ends with the NEXT I in line 3320.

The subroutine ends with the RETURN command in line 3330.

All four of the subroutines look similar, yet each will produce a different sound.

Line 9000 VTABs to line 21, the top of the text window, and then uses a prompted INPUT to hold the star field on the screen until the operator presses the RETURN key.

Line 9010 returns the computer to TEXT mode and clears the screen.

Line 9020 ENDs the program.

Answer To Assignment For Lesson Thirteen

Viewing Time 9:29

The assignment for Lesson Thirteen was to create a random access data file, and WRITE to and READ from the file.

The VideoSoft answer is a program to WRITE and READ data from a random access file called "CHECK.DATA".

```
10  REM :: RECORD CHECK USAGE
20  REM  SAVE CHECKFILE PGM
400  ONERR  GOTO 600
500  PRINT  CHR$ (4);"OPEN CHECK.DATA,L38"
510  PRINT  CHR$ (4);"READ CHECK.DATA ,R0"
520  INPUT C
530  GOTO 900
600  PRINT  CHR$ (4);"WRITE CHECK.DATA,R0"
610  PRINT 0
620 C = 0
900  PRINT  CHR$ (4);"CLOSE CHECK.DATA"
910  ONERR  GOTO 7000
1000  HOME : PRINT "--- CHECK DATA MENU ---"
1010  VTAB 5: PRINT "1 - WRITE DATA"
1020  PRINT : PRINT "2 - READ DATA"
1030  VTAB 10: INPUT "SELECT 1 OR 2, 0=END: ";S
1040  ON S + 1 GOTO 9000,2000,3000
1050  GOTO 1030
2000  HOME : PRINT "ENTER CHECK DESCRIPTION - RECORD ";C + 1
2010  INPUT "OR ENTER '/' TO END: ";C$
2020  IF C$ = "/" THEN 1000
2030  IF  LEN (C$) > 30 THEN  VTAB 23: PRINT "DESCRIPTION
      TOO LONG": GOSUB 8000 : GOTO 2000
2040  INPUT "HOW MUCH WAS THE CHECK FOR ? ";A
2050  IF A > 99999.99 THEN  VTAB 23: PRINT "CHECK AMOUNT
      TOO LARGE": GOSUB 8000 : GOTO 2000
2060  PRINT  CHR$ (4);"OPEN CHECK.DATA ,L38"
2070 C = C + 1
2080  PRINT  CHR$ (4);"WRITE CHECK.DATA ,R";C
2090  PRINT C$
2100  PRINT A
2110  PRINT  CHR$ (4);"CLOSE CHECK.DATA "
2120  VTAB 23: PRINT "CHECK RECORD WAS ADDED": GOSUB 8000:
      GOTO 2000
3000  HOME
3010  PRINT "ENTER RECORD NUMBER TO DISPLAY :1-";C
3020  INPUT "(0=END) ";R
3030  IF R = 0 THEN 1000
3040  PRINT  CHR$ (4);"OPEN CHECK.DATA ,L38"
3050  PRINT  CHR$ (4);"READ CHECK.DATA ,R";R
3060  INPUT C$
3070  INPUT A
3080  PRINT  CHR$ (4);"CLOSE CHECK.DATA"
3090  VTAB 5
3100  PRINT "THE CHECK WAS USED FOR ";C$
3110  PRINT "THE AMOUNT WAS $";A
3120  VTAB 9: INPUT "PRESS RETURN ";N$: GOTO 3000
7000  PRINT  CHR$ (4);"CLOSE CHECK.DATA"
```

```
7010  VTAB 23
7020  PRINT "RECORD NUMBER NOT FOUND"
7030  GOSUB 8000: GOTO 3000
8000  FOR I = 1 TO 740
8010  NEXT I
8020  RETURN
9000  PRINT  CHR$ (4);"OPEN CHECK.DATA ,L38"
9010  PRINT  CHR$ (4);"WRITE CHECK.DATA ,R0"
9020  PRINT C
9030  PRINT  CHR$ (4);"CLOSE CHECK.DATA"
9050  PRINT "PROGRAM ENDED"
9060  END
```

Lines 10-910 are the housekeeping section which determines how many records are in the file by reading the value of the accumulator stored in record number zero.

If the zero record does not exist, the ONERR GOTO command in line 400 sends the program to lines 600 through 620 which initializes the file by writing a zero in the control record.

Line 900 and 910 CLOSE the file and set the error trap for normal operations.

Lines 1000-1050 are the "MENU". The menu is used to allow the operator to choose whether to read, write, or end the program.

The ON-GOTO in line 1040 will send the program to the ending routine at line 9000 if zero is entered; the writing module at line 2000 if a one is entered; or the reading module at line 3000 if a two is entered.

Lines 2000-2120 comprise the file writing module.

Lines 2000 and 2010 instruct the operator to enter the check description for the next record or to enter a slash to end the writing section.

Line 2020 returns the program to the menu if a slash is entered.

Line 2030 checks the length of the description. If the length of C$ is greater than thirty, the message "DESCRIPTION TOO LONG" will be printed on line 23. A timing subroutine in line 8000 is executed, and the program returns to line 2000 to allow the operator to enter a shorter check description.

Line 2040 prompts the operator to enter the check amount.

Line 2050 error-checks the value of the amount that was entered. If the variable A is greater than ninety-nine thousand nine hundred ninety-nine and ninety-nine cents, the program VTABs 23 and, PRINTs the message "CHECK AMOUNT TOO LARGE". The subroutine at 8000 is again a timing loop. The program returns to line 2000 to allow for a corrected entry.

Line 2060 OPENs the "CHECK.DATA" file with a record length of thirty-eight; thirty characters for the name and eight for the amount.

Line 2070 advances the record number by adding one to C.

Line 2080 is the WRITE command.

Lines 2090 and 2100 PRINT the description and amount to the "CHECK.DATA" file.

Line 2110 is the required CLOSE command.

Line 2120 informs the operator that a record was added and returns the program to line 2000 so more data can be entered.

Lines 3000-3120 comprise the file reading module.

Lines 3000-3020 inform the operator to enter a valid record number, one through C, the number of records in the file.

If the operator enters zero, line 3030 returns the program to the menu at line 1000.

We prepare the file for use in line 3040 by OPENing the file.

In line 3050 we tell the computer to read record number "R".

Lines 3060 and 3070 INPUT the check description and amount from the file.

Line 3080 CLOSEs the file.

After VTABing to line 5 in line 3090, the program prints "THE CHECK WAS USED FOR", and the description in line 3100.

Line 3110 prints the check amount.

Line 3120 holds the check information on the screen with a dummy INPUT, then returns to line 3000 for the next entry.

Lines 7000 through 7030 constitute the error routine. The computer will only execute these lines if an error occurs.

Line 7000 CLOSEs the file because if an error occurs when the computer tries to read the record, the "ONERR GOTO" command in line 910 will make the computer jump immediately to line 7000, by-passing the CLOSE command in line 3080.

Lines 7010 and 7020 inform the operator that the requested record is not in the file.

Line 7030 executes the timing subroutine and returns the computer to line 3000.

Lines 8000 through 8020 are the timing subroutine.

Lines 9000 through 9060 are the ending routine.

Line 9000 OPENs the file.

In 9010, the WRITE command sets the file pointer to record number zero.

The current number of records is PRINTed to the file in line 9020.

The file is then CLOSEd and the program ended in lines 9030 through 9060.

Answers to Additional Assignments

LESSON ONE #1

```
20  REM  SAVE LES1A1
1000  HOME
1010  PRINT "THE HOME COMMAND"
1020  PRINT "CLEARS THE SCREEN."
1030  PRINT
1040  PRINT "THE PRINT COMMAND"
1050  PRINT "DISPLAYS MESSAGES"
1060  PRINT "ON THE SCREEN."
```

Line 1000 clears the screen with the HOME command.

Line 1010 prints the message "THE HOME COMMAND".

Line 1020 prints the message "CLEARS THE SCREEN".

Line 1030 uses a PRINT command without a literal to print a blank line.

Line 1040 prints "THE PRINT COMMAND".

Line 1050 prints "DISPLAYS MESSAGES".

Line 1060 prints "ON THE SCREEN".

LESSON ONE, #2

```
20  REM  SAVE LES1A2
1000  HOME
1010  PRINT "XX    XX"
1020  PRINT " XX XX"
1030  PRINT "  XXX"
1040  PRINT " XX XX"
1050  PRINT "XX    XX"
```

Line 1000 clears the screen with the HOME command.

Lines 1010 through 1050 print the lines of the "X".

Leading and trailing spaces always print.

LESSON TWO, #1

```
10  REM :: LESSON 2 ADDITIONAL 1
20  REM  SAVE LES2A1
1000  HOME
1010  INVERSE
1020  PRINT "FRANK"
1030  FLASH
1040  PRINT "FRANK"
1050  INVERSE
1060  PRINT "FRANK"
1070  FLASH
1080  PRINT "FRANK"
1090  INVERSE
1100  PRINT "FRANK"
1110  FLASH
1120  PRINT "FRANK"
1130  INVERSE
1140  PRINT "FRANK"
1150  FLASH
1160  PRINT "FRANK"
1170  INVERSE
1180  PRINT "FRANK"
1190  NORMAL
1200  END
```

Line 10 is a remark containing a brief description of the program.

Line 20 is the SAVE command for the program. Any name can be used.

Line 1000 clears the screen.

Line 1010 puts the computer in INVERSE print mode. This causes the PRINT command in line 1020 to display an INVERSE name on the screen.

Line 1030 puts the computer in flashing print mode. This causes the PRINT command in line 1040 to display a flashing name on the screen.

This sequence is repeated until four flashing names and five INVERSE names have been printed.

Then in line 1190 the computer is returned to NORMAL print mode.

The END command in line 1200 stops the program.

LESSON TWO, #2

```
10  REM :: LESSON 2 ADDITIONAL 2
20  REM  SAVE LES2A2
1000  HOME
1010  SPEED= 0
1020  PRINT "SPEED OF 0"
1030  FLASH
1040  PRINT "------------"
1050  INVERSE
1060  PRINT "------------"
1070  NORMAL
1080  SPEED= 50
1090  PRINT "SPEED OF 50"
1100  FLASH
1110  PRINT "------------"
1120  INVERSE
1130  PRINT "------------"
1140  NORMAL
1150  SPEED= 100
1160  PRINT "SPEED OF 100"
1170  FLASH
1180  PRINT "------------"
1190  INVERSE
1200  PRINT "------------"
1210  NORMAL
1220  SPEED= 150
1230  PRINT "SPEED OF 150"
1240  FLASH
1250  PRINT "------------"
1260  INVERSE
1270  PRINT "------------"
1280  NORMAL
1290  SPEED= 200
1300  PRINT "SPEED OF 200"
1310  FLASH
1320  PRINT "------------"
1330  INVERSE
1340  PRINT "------------"
1350  NORMAL
1360  SPEED= 255
1370  PRINT "SPEED OF 255"
1380  FLASH
1390  PRINT "------------"
1400  INVERSE
1410  PRINT "------------"
1420  NORMAL
9000  END
```

Line 10 and 20 are the opening remarks.

Line 1000 clears the screen.

Line 1010 sets the print rate to zero.

Line 1020 prints the message to the operator, "SPEED OF 0".

Line 1030 puts the computer in flashing print mode.

Line 1040 instructs the computer to print a line of twelve dashes. These dashes will print in flashing mode.

In line 1050 we turn on INVERSE print mode using the INVERSE command.

Line 1060 than prints another line of twelve dashes. This time the dashes print in INVERSE mode.

In line 1070 we begin the next cycle by returning the computer to NORMAL print mode.

Then in line 1080 we reset the speed to fifty and perform the same operations we did with a speed of zero.

This continues for each of the six print speeds.

Then line 1420 returns the computer to NORMAL print mode.

Line 9000 then ends the program.

LESSON THREE, #1

```
10  REM :: LESSON 3 ADDITIONAL 1
20  REM SAVE LES3A1
500 M1$ = "YOUR NEIGHBOR'S NAME IS"
510 M2$ = "YOUR DOCTOR'S NAME IS"
1000  HOME
1010  INPUT "WHAT IS YOUR DOCTOR'S NAME ? ";D$
1020  INPUT "WHAT IS YOUR NEIGHBOR'S NAME ? ";N$
1030  HOME
1040  PRINT M1$
1050  FLASH
1060  PRINT N$
1070  NORMAL
1080  PRINT M2$
1090  FLASH
1100  PRINT D$
1110  NORMAL
1120  END
```

Lines 10 and 20 are the opening remarks.

In line 500 the string literal "YOUR NEIGHBOR'S NAME IS" is assigned to the variable M1$.

In line 510 the string literal "YOUR DOCTOR'S NAME IS" is assigned to the variable M2$.

Line 1000 clears the screen.

Lines 1010 and 1020 are prompted inputs which allow the operator to enter the doctor's and neighbor's names.

The doctor's name is stored in the variable D$.

The neighbor's name is stored in the variable N$.

Line 1030 clears the screen.

Line 1040 prints the first string literal, M1$, on the screen.

Line 1050 turns on flashing print mode.

In line 1060 the value of N$, which was entered in response to the question, "WHAT IS YOUR NEIGHBOR'S NAME ? ", is printed.

Line 1070 returns the computer to normal print mode.

Line 1080 prints the second string literal, M2$, on the screen.

Line 1090 switches the computer to flashing mode.

Line 1100 prints the doctor's name stored in D$.

Line 1110 returns the computer to normal print mode.

Line 1120 ends the program.

LESSON THREE, #2

```
10  REM :: LESSON 3 ADDITIONAL 2
20  REM SAVE LES3A2
500 M1$ = "I ALSO HAVE A FRIEND NAMED"
510 M2$ = "I WOULD ALSO LIKE A"
520 M3$ = "FOR A PET"
530 M4$ = "IS ALSO MY FAVORITE SEASON"
1000  HOME
1010  INPUT "WHAT IS YOUR FRIEND'S NAME ? ";F$
1020  PRINT
1030  PRINT M1$
1040  PRINT F$
1050  PRINT
1060  INPUT "WHAT ANIMAL WOULD YOU LIKE FOR A PET ? ";P$
1070  PRINT
1080  PRINT M2$
1090  PRINT P$
1100  PRINT M3$
1110  PRINT
1120  INPUT "WHAT IS YOUR FAVORITE SEASON ? ";S$
1130  PRINT
1140  PRINT S$
1150  PRINT M4$
9000  END
```

Lines 10 and 20 are the opening remarks.

Line 500 assigns the string literal "I ALSO HAVE A FRIEND NAMED" to the variable M1$.

Line 510 assigns the string literal "I WOULD ALSO LIKE A" to the variable M2$.

Line 520 assigns the string literal "FOR A PET" to the variable M3$.

Line 530 assigns the string literal "IS ALSO MY FAVORITE SEASON" to the variable M4$.

Line 1000 clears the screen.

Line 1010 is a prompted input which displays the question "WHAT IS YOUR FRIEND'S NAME ? " on the CRT and stores the operator's answer in the variable F$.

Line 1020 prints a blank line on the screen.

Line 1030 prints the string literal that is stored in the variable M1$.

Line 1040 prints the friend's name that is stored in the variable F$.

Line 1050 prints a blank line to make the screen easier to read.

Line 1060 is a prompted input which displays the message "WHAT ANIMAL WOULD YOU LIKE FOR A PET ? " and stores the operator's answer in the variable P$.

Line 1070 prints a blank line.

Line 1080 prints the string literal that is stored in M2$.

Line 1090 prints the entry that is stored in the variable P$.

Line 1100 prints the string literal that is stored in M3$.

Line 1110 prints a blank line.

Line 1120 prompts with the message "WHAT IS YOUR FAVORITE SEASON? " and stores the operator's answer in the variable S$.

Line 1130 prints a blank line.

Line 1140 prints the value that is stored in the variable S$.

Line 1150 prints the string literal that is stored in the variable M4$.

Line 9000 ends the program.

LESSON FOUR, #1

```
10  REM  :: LESSON 4 ADDITIONAL 1
20  REM   SAVE LES4A1
1000  HOME
1010  VTAB 20
1020  INPUT "ENTER YOUR AGE ";A
1030  VTAB 2
1040  PRINT "YOU ARE ";A;" YEARS OLD"
1050  VTAB 25
1060  INPUT "ENTER YOUR NAME ";N
1070  VTAB 3
1080  PRINT "YOUR NAME IS ";N$
1090  VTAB 20
1100  INPUT "ENTER YOUR FAVORITE COLOR ";C$
1110 XTAB4
1120  PRINT "YOUR FAVORITE COLOR IS ";C$
1130  END
```

ERROR ONE:
Line 1050 instructs the computer to position the cursor on screen line twenty-five with the command VTAB 25.

Because there are only twenty-four screen lines this is an invalid command.

VTAB 25 should be changed to VTAB followed by a number from one to twenty-four.

ERROR TWO:
Line 1060 instructs the operator to enter a name.

Because the variable used in the INPUT is a numeric variable, a name cannot be entered.

Adding a dollar sign to the variable N will correct the error.

ERROR THREE:
Line 1110 uses an XTAB command which is not a valid command.

Change XTAB to VTAB or HTAB to correct the error.

LESSON FOUR, #2

```
10  REM  :: LESSON 4 ADDITIONAL 2
20  REM  SAVE LES4A2
1000  HOME
1010  VTAB 2
1020  PRINT  TAB( 12);"VTAB 2,TAB 12"
1030  VTAB 4
1040  HTAB 4
1050  PRINT "VTAB 4,HTAB 4"
1060  VTAB 4
1070  HTAB 22
1080  PRINT "VTAB 4,HTAB 22"
1090  VTAB 6
1100  PRINT "ZONE 1","ZONE 2","ZONE 3"
1110  VTAB 8
1120  PRINT "XXXXXXXXXXXXX";
1130  PRINT "-------------";
1140  PRINT "XXXXXXXXXXXXX"
1150  END
```

Line 1010 positions the print cursor on screen line 2.

Line 1020 prints the message "VTAB 2, TAB 12" after tabbing to the twelfth screen position.

The VTAB 4 command in line 1030 positions the cursor on screen line four.

HTAB 4 in line 1040 positions the cursor in column four.

Lines 1030 and 1040 cause the message in line 1050, "VTAB 4, HTAB 4", to print at screen line four, column four.

Lines 1060 and 1070 cause the message in line 1080, "VTAB 4, HTAB 22", to print beginning on screen line four, column twenty-two.

Line 1090 positions the cursor on line six of the screen.

Line 1100 prints each of the three messages in a separate print zone.

Line 1110 positions the cursor on screen line eight.

Line 1120 prints thirteen X's and uses a semicolon to hold the print cursor at the end of the literal.

Then line 1130 prints thirteen dashes followed by a semicolon. These dashes will follow immediately after the thirteen X's that were printed in line 1120.

Line 1140 prints another thirteen X's. These will immediately follow the thirteen dashes because of the semicolon following the quotation marks in line 1130.

Line 1150 ends the program.

LESSON FIVE, #1

```
10  REM  :: LESSON 5 ADDITIONAL 1
20  REM  SAVE LES5A1
1000  HOME
1010  VTAB 1
1020  INPUT "ENTER A NAME OR (STOP) TO END ";N$
1030  IF N$ = "STOP" THEN 9000
1040  HOME
1050  VTAB 12
1060  HTAB 20
1070  PRINT N$
1080  VTAB 16
1090  HTAB 5
1100  PRINT N$
1110  VTAB 5
1120  PRINT N$,N$
1130  VTAB 20
1140  PRINT N$;N$;N$
1150  GOTO 1010
9000  END
```

Lines 10 and 20 are the opening remarks.

Line 1000 clears the screen and 1010 positions the cursor on screen line one.

The prompted INPUT in line 1020 allows the operator to enter a name and stores the name in the variable N$.

In line 1030 if the name entered was "STOP", then the computer is instructed to GOTO line 9000.

Line 1040 clears the screen.

The VTAB 12 and HTAB 20 commands in lines 1050 and 1060 position the cursor for the first printing of the name.

Line 1070 prints the name.

Line 1080 positions the cursor on screen line sixteen.

Line 1090 positions the cursor at position five of screen line sixteen.

Line 1100 prints the name on line sixteen at position five.

Line 1110 positions the cursor on screen line five.

In line 1120 the name is printed in print zones one and two.

Line 1130 positions the cursor on line twenty.

Line 1140 prints the name three times on the same line.

The GOTO command in line 1150 sends the computer back to line 1010 to begin the loop again.

Line 9000 ends the program.

LESSON FIVE, #2

```
10  REM  :: LESSON 5 ADDITIONAL 2
20  REM  SAVE LES5A2
1000  HOME
1010  PRINT "COULD YOU ANSWER A FEW QUESTIONS"
1020  INPUT "FOR ME ? ";A$
1030  IF A$ = "NO" THEN 9000
1040  IF A$ = "YES" THEN  PRINT "THANK YOU"
1050  PRINT
1060  INPUT "IS TODAY TUESDAY ? ";B$
1070  IF B$ = "NO" THEN  PRINT "I WAS SURE IT WAS TUESDAY"
1080  IF B$ = "YES" THEN  PRINT "I KNEW IT ALL THE TIME"
1090  PRINT
1100  INPUT "ARE ELEPHANTS BIG AND GREY ? ";C$
1110  IF C$ = "NO" THEN  PRINT "YOU ARE TRYING TO TRICK ME"
1120  IF C$ = "YES" THEN  PRINT "I THOUGHT SO"
1130  PRINT
1140  IF C$ = "YES" THEN  PRINT "THANKS FOR THE INFORMATION"
9000  END
```

Line 1000 clears the screen.

Line 1010 prints the first half of question one.

Line 1020 prints the second half of question one and allows the operator to enter the answer.

Line 1030 sends the computer to the program end, line 9000, if "NO" was entered.

If "YES" was entered in response to the first question, line 1040 prints the message "THANK YOU".

Line 1050 prints a blank line to separate the questions.

Line 1060 asks the second question, "IS TODAY TUESDAY ? ", and allows the operator to enter the answer.

If the operator entered the answer "NO", the message "I WAS SURE IT WAS TUESDAY" will be printed in line 1070.

The message "I KNEW IT ALL THE TIME" will be printed if the operator responded "YES" to the question.

Line 1090 prints a blank line to make the dialogue more readable.

The last question, "ARE ELEPHANTS BIG AND GREY ? ", is asked in line 1100.

The answer to the question is stored in the variable C$.

If "NO" is entered, line 1110 will print the message "YOU ARE TRYING TO TRICK ME".

Line 1120 will print the message "I THOUGHT SO" if C$ equals "YES".

Line 1130 prints a blank line.

The operator is thanked for the information if the last question was answered with "YES".

Line 9000 ends the program.

LESSON SIX, #1

```
10  REM :: LESSON 6 ADDITIONAL 1
20  REM SAVE LES6A1
990  REM :: INSTRUCTIONS
1000  HOME
1010  PRINT "IN THIS PROGRAM YOU WILL BE ALLOWED TO"
1020  PRINT "ENTER A PHRASE. YOU CAN DEFINE THE MODE"
1030  PRINT "AND SPEED FOR THE PHRASE TO BE PRINTED"
1040  PRINT "ON THE SCREEN"
1050  VTAB 10
1060  PRINT "ENTER THE WORD 'STOP' TO END."
1070  VTAB 20
1080  INPUT "PRESS (RETURN) TO BEGIN ";S$
1990  REM :: INPUT SECTION
2000  HOME
2010  INPUT "ENTER THE PHRASE ";W$
2020  IF W$ = "STOP" THEN 9000
2030  PRINT "1=NORMAL 2=FLASH 3=INVERSE"
2040  INPUT "SELECT PRINT MODE ";M
2050  IF M < 1 OR M > 3 THEN 2030
2060  PRINT "CHOOSE SPEED FOR PRINT (0-255)"
2070  INPUT "ENTER SPEED ";S
2080  IF S < 0 OR S > 255 THEN 2060
2990  REM :: PRINT SECTION
3000  HOME
3010  IF M = 1 THEN  NORMAL
3020  IF M = 2 THEN  FLASH
3030  IF M = 3 THEN  INVERSE
3040  SPEED= S
3050  VTAB 12
3060  HTAB 10
3070  PRINT W$
3080  NORMAL
3090  SPEED= 255
3100  VTAB 20
3110  HTAB 1
3120  INPUT "PRESS (RETURN) ";X$
3130  GOTO 2000
8990  REM :: END OF JOB
9000  HOME
9010  END
```

MODULE ONE: HOUSEKEEPING (Lines 10-20)

Lines 10 and 20 are the standard opening remarks.

MODULE TWO: INSTRUCTIONS (Lines 1000-1080)

The HOME command in line 1000 clears the screen.

Lines 1010 through 1060 PRINT instructions to the operator.

Line 1070 positions the cursor at line 20.

Line 1080 uses a dummy INPUT to hold the instructions on the screen until the operator presses the "RETURN" key.

MODULE THREE: INPUT SECTION (Lines 2000-2080)

Line 2000 clears the screen.

The operator is allowed to enter the phrase to be printed in line 2010.

If "STOP" was entered by the operator, line 2020 will send the computer to line 9000.

Line 2030 prints a message to show the operator the available print options.

Line 2040 allows the operator to select the print mode.

The print mode entry is error checked by line 2050.

Line 2060 informs the operator of the available SPEED selections.

Line 2070 allows the operator to enter the printing speed and stores the number entered in the variable S.

Line 2080 returns the computer to line 2060 if an invalid SPEED entry was made.

MODULE FOUR: PRINT SECTION (Lines 3000-3130)

Line 3000 clears the screen.

Lines 3010 through 3030 use a series of three IF-THENs to switch to the print mode that was selected by the operator.

Line 3040 sets the print speed to the value entered by the operator.

Line 3050 and 3060 position the cursor on line 12, position 10.

The phrase that was entered is printed in line 3070.

Line 3080 returns the computer to NORMAL print mode.

Line 3090 returns the computer to normal print speed, 255.

Lines 3100 and 3110 position the print cursor near the bottom of the screen to line 20, position 1.

The prompted INPUT in line 3120 holds the printed phrase on the screen for the operator to see.

Line 3130 returns the computer to line 2000 so the operator can make another entry.

MODULE FIVE: END OF JOB (Lines 9000-9010)

Line 9000 clears the screen.

Line 9010 ends the program.

LESSON SIX, #2

Program Outline & Logic Paragraphs

MODULE ONE: HOUSEKEEPING

Standard opening remarks. Assign hidden creature location: screen line to "HL" (range 1 to 22) and column to "HC" (range 1 to 40).

MODULE TWO: INSTRUCTIONS

Begin in line 1000. Inform operator of range of hiding places and how to enter guess or zero to end. Use dummy INPUT to hold on screen for operator.

MODULE THREE: PLAY OF GAME

Begin in line 2000. Use multiple input statement to accept line and column from operator. Store line in "L" and column in "C". Prompt at screen line 23. Use IF-THENs with OR to test for zeros, to test for proper line and column ranges, and for successful guess.

If guess is successful, go to "Player Wins" section.

If guess is not successful print asterisk in location guessed and allow operator to guess again.

MODULE FOUR: PLAYER WINS

Begin in line 3000. PRINT flashing asterisk in correct position, and return to normal print mode.

MODULE FIVE: END OF GAME

Begin in line 9000. End game without disrupting CRT display so player can see winning position.

```
10  REM :: LESSON 6 ADDITIONAL 2
20  REM SAVE LES6A2
30  REM :: HOUSEKEEPING
500 HL = 21
510 HC = 6
990  REM :: INSTRUCTIONS
1000  HOME
1010  PRINT "THERE IS A CREATURE HIDING SOMEWHERE ON"
1020  PRINT "A 22 LINE BY 40 COLUMN SCREEN.  ENTER"
1030  PRINT "LINE AND COLUMN UNTIL YOU FIND IT OR"
1040  PRINT "ENTER A ZERO TO END."
1050  VTAB 20
1060  INPUT "PRESS (RETURN) TO BEGIN PLAY";N$
1070  HOME
1990  REM :: PLAY OF GAME
2000  VTAB 23: HTAB 1: INPUT "ENTER LINE,COLUMN ";L,C
2010  IF L = 0 OR C = 0 THEN 9000
2020  IF L < 1 OR L > 22 THEN  PRINT  CHR$ (7);: GOTO 2000
2030  IF C < 1 OR C > 40 THEN  PRINT  CHR$ (7);: GOTO 2000
2040  IF C = HC AND L = HL THEN 3000
2050  VTAB L
2060  HTAB C
2070  PRINT "*";
2080  GOTO 2000
2990  REM :: PLAYER WINS
3000  VTAB L
3010  HTAB C
3020  FLASH
3030  PRINT "*";
3040  NORMAL
8990  REM :: END OF GAME
9000  VTAB 1
9010  HTAB 1
9020  END
```

MODULE ONE: HOUSEKEEPING (Lines 10-510)

Lines 10 and 20 document the program name and the SAVE command.

In lines 500 and 510 the hidden line and column numbers are assigned to "HL" and "HC".

MODULE TWO: INSTRUCTIONS (Lines 1000-1070)

The screen is cleared in line 1000.

The instructions to the operator are printed in lines 1010 through 1040.

Line 1050 positions the cursor near the bottom of the screen on line 20.

The dummy INPUT in line 1060 holds the instructions on the screen until the operator presses the "RETURN" key.

Line 1070 clears the screen.

MODULE THREE: PLAY OF GAME (Lines 2000-2080)

Line 2000 positions the cursor on screen line 23 and allows the operator to enter a guess using a multiple INPUT command.

If the line or column entered is zero, line 2010 will instruct the computer to GOTO line 9000 for the ending section.

Line 2020 checks the line entry. If an invalid line is entered, the program will print CHR$(7), the bell, and GOTO line 2000 for another entry.

Line 2030 checks the column entry. If an invalid column number is entered, the program prints CHR$(7) and returns to line 2000 for another entry.

Line 2040 compares the entries made by the operator to the values defined for the hidden row and column. If both the line and column entries match, the computer is instructed to GOTO line 3000.

The cursor is positioned to the line entered by the operator in line 2050.

Line 2060 positions the cursor to the column entered by the operator.

Line 2070 prints an asterisk in the position selected by the operator.

The GOTO command in line 2080 returns the computer to line 2000 to allow another guess to be entered.

MODULE FOUR: PLAYER WINS (Lines 3000-3040)

In this module the cursor is first positioned to the line and column of the operator's entry by lines 3000 and 3010.

Line 3020 switches the computer to FLASH print mode.

Line 3030 prints a flashing asterisk in the position of the hidden creature.

Line 3040 returns the computer to NORMAL print mode.

MODULE FIVE: END OF GAME (Lines 9000-9020)

Lines 9000 and 9010 position the cursor at the top of the screen. This is done to keep the position of the flashing asterisk from changing position when the program ends.

Line 9020 ends the program.

LESSON SEVEN, #1

```
10  REM :: LESSON 7 ADDITIONAL 1
20  REM  SAVE LES7A1
500 A = 0
510 B = 100
1000  HOME
1010 A = A + 1
1020 B = B - 1
1030  PRINT "A ";A,"B ";B
1040  IF A = B THEN  GOTO 9000
1050  GOTO 1010
9000  END
```

Line 500 defines the starting value of the first accumulator, A, as zero.

Line 510 defines the starting value of the second accumulator, B, as one hundred.

The HOME command in line 1000 clears the screen.

Line 1010 adds one to the first accumulator.

Line 1020 subtracts one from the second accumulator.

Line 1030 prints the accumulators.

If the accumulators are equal, line 1040 sends the computer to line 9000.

Line 1050 returns the computer to line 1010 for another pass of the loop.

Line 9000 ends the program.

WHY PROGRAM DOES NOT WORK IF SECOND ACCUMULATOR STARTS AT 99.

If the first accumulator starts at zero and the second accumulator starts at 99, the two accumulators will never be equal to each other.

LESSON SEVEN, #2

```
10  REM :: LESSON 7 ADDITIONAL 2
20  REM   SAVE LES7A2
1000  HOME
1010  FOR I = 1 TO 20
1017  REM  ::::::::::::
1018  REM  :: PRINT UP SCREEN
1019  REM  ::::::::::::
1020 J = 20
1030  FOR K = 2 TO 20 STEP 2
1040 J = J - 1
1050  VTAB J
1060  HTAB K
1070  PRINT "**"
1080  NEXT K
1087  REM  ::::::::::::
1088  REM  :: PRINT DOWN SCREEN
1089  REM  ::::::::::::
1090  FOR K = 22 TO 38 STEP 2
1100 J = J + 1
1110  VTAB J
1120  HTAB K
1130  PRINT "**"
1140  NEXT K
1150  VTAB 24
1160  PRINT
1170  NEXT I
9000  END
```

Lines 10 and 20 are the opening remarks.

Line 1000 clears the screen.

In line 1010 we start the main program loop. This loop will cause the arrow to be printed twenty times.

Line 1020 assigns a beginning value of twenty to the accumulator J. J is used to track the vertical print position.

Line 1030 starts a loop that is used to increment the horizontal print position. In this loop the first half of the arrow will be printed.

Line 1040 subtracts one from J for each pass of the loop.

VTAB J in line 1050 positions the print cursor on screen line J.

HTAB K in line 1060 positions the print cursor to the horizontal position K.

Line 1070 prints two asterisks.

Line 1080 completes the first K loop.

Line 1090 begins the second K loop. This loop will print the second half of the arrow.

Line 1100 adds one to the value of J, causing the asterisks to advance down the screen.

Line 1110 positions the print cursor to print line J.

Line 1120 positions the print cursor at position K of screen line J.

The two asterisks are printed in line 1130.

Line 1140 ends the second K loop.

Line 1150 positions the print cursor on screen line twenty-four.

Line 1160 prints a blank space. This makes the arrow scroll up the screen.

The I loop that was started at the beginning of the program is completed by line 1170.

Line 9000 ends the program after the twenty passes of the I loop are completed.

LESSON EIGHT, #1

```
10  REM :: LESSON 8 ADDITIONAL 1
20  REM  SAVE LES8A1
1000  GOTO 8000
1010  PRINT "PLEASE SELECT..."
1020  PRINT " 1 - NAME...: ";N$
1030  PRINT " 2 - ADDRESS: ";A$
1040  PRINT " 3 - CITY...: ";C$
1050  PRINT
1060  NORMAL
1070  PRINT "ENTER 1 - 3 OR 0 TO END: ";S
1080  IF S = 0 THEN 5000
1090  AT S GOTO 1200,2000,3000
1100  GOTO 1070
1200  GOSUB 8000
1210  PRINT "CURRENT NAME IS ";N$
1220  INPUT "ENTER CORRECT NAME: ";N$
1230  GOTO 1000
2000  GOSUB 8000
```

```
2010  PRINT "CURRENT ADDRESS IS ";A$
2020  INPUT "ENTER CORRECT ADDRESS: ";A$
2030  GOTO 1000
3000  GOSUB 8000
3010  PRINT "CURRENT CITY IS ";C$
3020  INPUT "ENTER CORRECT CITY: ";C$
3030  GOTO 1000
5000  GOSUB 8000
5010  PRINT "PLEASE VERIFY ENTRIES"
5020  PRINT " NAME...: ";N$
5030  PRINT " ADDRESS: ";A$
5040  PRINT " CITY...: ";C$
5050  PRINT
5060  INPUT "ENTER Y OR N: ";AN$
5070  IF AN$ = "N" THEN 1000
8000  HOME
8010  VTAB 8
8020  INVERSE
8030  RETURN
9000  GOSUB 8000
9010  PRINT "PROGRAM ENDED"
9020  END
```

ERROR ONE:
Line 1000 uses a GOTO to execute a subroutine. Replace the GOTO with GOSUB. GOTO and GOSUB are not interchangeable.

ERROR TWO:
Line 1070 uses a PRINT command instead of an INPUT command. Replace the PRINT command with an INPUT command.

ERROR THREE:
Line 1090 uses the word "AT" instead of the command ON. Replace the word "AT" with the ON command.

ERROR FOUR:
When ending the program, the computer is allowed to execute the line of the subroutine at 8000 rather than by-passing the subroutine with a GOTO. Add a line after line 5070 which instructs the computer to GOTO 9000.

With subroutines remember: (1) a subroutine must be executed with a GOSUB command; (2) subroutines must be "protected" with GOTO commands which instruct the computer to by-pass the subroutine; (3) GOSUB and GOTO are not interchangeable.

ERROR FIVE:
The final error is in the ending section beginning at 9000. The program fails to return the computer to NORMAL print mode. (The 8000 subroutine switches to INVERSE mode.) Use the command NORMAL before ending the program.

LESSON EIGHT, #2

```
10  REM :: LESSON 8 ADDITIONAL 2
20  REM  SAVE LES8A2
1000  HOME
1010  PRINT "ENTER A WORD FROM 1-5 CHARACTERS LONG"
1020  INPUT "OR ENTER 'STOP' TO END ";W$
1030  IF W$ = "STOP" THEN 9000
1040  SPEED= 150
```

```
1997  REM  ::::::::::::::::
1998  REM  ::EXECUTE SUBROUTINES
1999  REM  ::::::::::::::::
2000  FOR J = 1 TO 4
2010  HOME
2020  ON J GOSUB 3000,5000,4000,6000
2030  NEXT J
2040  SPEED= 255
2050  GOTO 1000
2997  REM  ::::::::::::::
2998  REM  :: LEFT TO RIGHT
2999  REM  ::::::::::::::
3000  FOR I = 6 TO 30 STEP 6
3010  VTAB 10
3020  HTAB I
3030  PRINT W$
3040  HTAB I
3050  PRINT W$
3060  HTAB I
3070  PRINT W$
3080  NEXT I
3090  RETURN
3997  REM  ::::::::::::::
3998  REM  :: RIGHT TO LEFT
3999  REM  ::::::::::::::
4000  FOR I = 32 TO 8 STEP  - 6
4010  VTAB 10
4020  HTAB I
4030  PRINT W$
4040  HTAB I
4050  PRINT W$
4060  HTAB I
4070  PRINT W$
4080  NEXT I
4090  RETURN
4997  REM  ::::::::::::::::::
4998  REM  :: TOP TO BOTTOM
4999  REM  ::::::::::::::::::
5000  FOR I = 1 TO 23
5010  HTAB 18
5020  PRINT W$
5030  NEXT I
5040  RETURN
5997  REM  ::::::::::::::::
5998  REM  :: BOTTOM TO TOP
5999  REM  ::::::::::::::::
6000  FOR I = 23 TO 1 STEP  - 1
6010  VTAB I
6020  HTAB 18
6030  PRINT W$
6040  NEXT I
6050  RETURN
8997  REM  ::::::::::::::::::
8998  REM  :: END PROGRAM
8999  REM  ::::::::::::::::::
9000  HOME
9010  END
```

Lines 10 and 20 are the opening remarks.

Line 1000 clears the screen.

Lines 1010 and 1020 prompt the operator to enter a word from 1 to 5 characters in length and store the entered word in the variable W$.

Line 1030 sends the computer to line 9000 for the ending routine if the word "STOP" is entered.

Line 1040 sets the computer's print speed to 150.

Lines 2000 through 2050 are the control section of the program.

Line 2000 defines the program loop.

Line 2010 clears the screen before each of the PRINT subroutines is executed.

Line 2020 executes the subroutine beginning at line 3000, 5000, 4000 or 6000 depending on the value of J.

Line 2030 is the end of the main program loop.

After the word is printed in the four formats, line 2040 sets the print rate to the maximum, and line 2050 returns the computer to line 1000.

Lines 3000-3090 print the word from left to right on the screen.

Line 3000 defines a PRINT loop that will execute five times increasing the value of I by six each time.

Line 3010 positions the cursor on screen line 10. This causes the word to be printed near the middle of the screen.

Line 3020 positions the cursor at position I. This causes the first row of words to print starting at position 6, the second row at 12, etc.

Line 3030 prints the word.

Line 3040 repositions the cursor on the next line at the same horizontal position.

Line 3050 again prints the word.

Line 3060 repositions the cursor.

Line 3070 prints the word on screen line 12.

Line 3080 completes the left to right print loop.

The RETURN command in line 3090 completes the subroutine.

Lines 4000-4090 are the right to left PRINT subroutine.

Line 4000 defines the print loop beginning at 32 and decreases the value of I until I equals eight. The STEP-6 command causes the value of I to be decreased by six for each pass of the loop.

Line 4010 positions the cursor on screen line 10.

Lines 4020 through 4070 print the word three times, using the horizontal position of I.

Line 4080 completes the print loop.

Line 4090 ends the subroutine.

Lines 5000 through 5040 print the word from the top of the screen to the bottom.

Line 5000 defines the loop.

Line 5010 positions the cursor.

Line 5020 prints the word.

Line 5030 completes the loop.

Line 5040 ends the subroutine.

Lines 6000 through 6050 print the word from the bottom of the screen to the top.

Line 6000 defines a loop in which the loop variable starts at 23 and counts backwards to 1.

Lines 6010 and 6020 position the cursor.

Line 6030 prints the word.

Lines 6040 and 6050 end the loop and the subroutine.

Line 9000 clears the screen.

Line 9010 ends the program.

LESSON NINE, #1

```
10  REM :: LESSON 9 ADDITIONAL 1
20  REM  SAVE LES9A1
50  DIM P$(1,8)
500  DATA PITCHER,FIRST BASE,SECOND BASE,THIRD BASE,SHORTSTOP,
     LEFT FIELD,RIGHT FIELD,CENTER FIELD,CATCHER
600  FOR I = 0 TO 8
610  READ P$(0,I)
620  NEXT I
997  REM  ::::::::::::::
998  REM  :: GET PLAYER NAMES
999  REM  ::::::::::::::
1000  FOR I = 0 TO 8
1010  HOME
1020  PRINT "WHO WILL PLAY ";P$(0,I);
1030  INPUT P$(1,I)
1040  NEXT I
1997  REM  ::::::::::::::
1998  REM  :: PRINT FIELD
1999  REM  ::::::::::::::
2000  HOME
2010  FOR I = 0 TO 8
2020  PRINT P$(1,I);" WILL PLAY ";P$(0,I)
2030  PRINT
2040  NEXT I
9000  END
```

Lines 10 and 20 are the opening remarks.

Line 50 dimensions a two-dimensional array with two rows and nine elements in each row.

Line 500 is a DATA statement which holds the names of each of the team positions.

Lines 600 through 620 form a loop that is used to READ the position names into the first row of the array.

Line 1000 starts our data entry loop.

Line 1010 clears the screen at the beginning of each pass of the loop.

Line 1020 asks the coach the question "WHO WILL PLAY ", and the name of the position. The semicolon at the end of the PRINT statement causes the INPUT to occur on the same line as the PRINT.

Line 1030 allows the coach to enter each of the names.

The data entry loop is completed by the NEXT command in line 1040.

Line 2000 clears the screen.

Line 2010 starts the PRINT loop.

For each pass of the loop line 2020 prints the name of the player and that player's assigned position.

A blank line is printed between each printed line to make the list more readable.

Line 2040 is the end of the print loop.

Line 9000 ends the program.

LESSON NINE, #2

```
10  REM :: LESSON 9 ADDITIONAL 2
20  REM  SAVE LES9A2
50  DIM G(20)
997  REM  ::::::::::::::::
998  REM  :: INSTRUCTIONS
999  REM  ::::::::::::::::
1000  HOME
1010  PRINT "THIS PROGRAM WILL ALLOW YOU TO ENTER UP"
1020  PRINT "TO TWENTY GRADES TO BE AVERAGED"
1030  PRINT
1040  PRINT "TO OPERATE THE PROGRAM ENTER THE"
1050  PRINT "GRADES TO BE AVERAGED ONE AT A TIME."
1060  PRINT "WHEN THE LAST GRADE HAS BEEN ENTERED,"
1070  PRINT "ENTER -1 AND THE AVERAGE WILL BE"
1080  PRINT "DISPLAYED ON THE SCREEN."
1090  VTAB 20
1100  INPUT "PRESS (RETURN) TO BEGIN";N$
1997  REM  ::::::::::::::
1998  REM  :: ENTER GRADES
1999  REM  ::::::::::::::
2000  FOR GN = 1 TO 20
2010  HOME
2020  PRINT "ENTER GRADE # ";GN
2030  INPUT "OR -1 TO END ";G(GN)
```

```
2040  IF G(GN) =  - 1 THEN 3000
2050 T = T + G(GN)
2060  NEXT GN
2997  REM  ::::::::::::::::
2998  REM  :: CALCULATE AVERAGE
2999  REM  ::::::::::::::::
3000 A = T / (GN - 1)
3997  REM  ::::::::::::::::
3998  REM  :: PRINT AVERAGE
3999  REM  ::::::::::::::::
4000  HOME
4010  VTAB 10
4020  HTAB 16
4030  PRINT A
4040  VTAB 20
4050  HTAB 1
4060  INPUT "YOUR AVERAGE - PRESS (RETURN) TO END";N$
9000  HOME
9010  END
```

Lines 10 and 20 are the opening remarks.

Line 50 dimensions the G array to hold the grades.

Line 1000 clears the screen.

The instructions to the operator are printed in lines 1010 through 1080.

Line 1090 positions the cursor near the bottom of the screen on line 20.

Line 1100 is a dummy INPUT that is used to hold the instructions on the CRT.

Line 2000 starts the grade entry loop using the variable GN as the loop variable.

For each pass of the loop, line 2010 clears the screen.

Line 2020 prints the grade number that can currently be entered.

The operator is allowed to enter a grade or enter a negative one in line 2030.

If a negative one is entered, line 2040 sends the computer to the average calculation section.

Line 2050 adds each of the grades to the total as they are entered. (Notice the use of an accumulator to total a series of numbers.)

Line 2060 is the end of the entry loop.

The average is calculated in line 3000 using the equation AVERAGE = TOTAL / (GRADE NUMBER -1).

One is subtracted from the grade number because if the student enters a negative one to end the entry loop, the last entry is the negative one and should be ignored.

This also demonstrates another characteristic of a FOR-NEXT loop: if the program exits from the loop before its completion, the loop variable will retain its value.

Line 4000 clears the screen for the printing of the average.

Line 4010 and 4020 position the print cursor near the middle of the CRT on line 10, position 16.

Line 4030 prints the average.

Lines 4040 and 4050 position the cursor near the bottom of the CRT on line 20, position one.

Line 4060 holds the average on the screen until the operator presses the "RETURN" key to end the program.

Line 9000 clears the screen.

Line 9010 ends the program.

LESSON TEN, #1

```
10  REM : LESSON 10 ADDITIONAL 1
20  REM  SAVE LES10A1
1000  HOME
1010  PRINT "ENTER A PHRASE FROM 1-19 CHARACTERS IN"
1020  INPUT "LENGTH ";W$
1030  IF  LEN (W$) > 19 THEN 1000
2000  HOME
2010  FOR I = 1 TO  LEN (W$)
2020 W2$ =  LEFT$ (W$,I)
2030 W3$ =  RIGHT$ (W$,I)
2040  PRINT  TAB( 1);W2$;
2050  PRINT  TAB( 40 -  LEN (W3$));W3$
2060  NEXT I
3000  VTAB 21
3010  INPUT "ANOTHER PHRASE (Y/N) ";YN$
3020  IF YN$ = "Y" THEN 1000
3030  IF YN$ <  > "N" THEN 3000
9000  HOME
9010  END
```

The opening remarks are contained in lines 10 and 20.

Line 1000 clears the screen.

The first part of the operator's prompt is printed in line 1010.

Line 1020 prints the second part of the prompt and allows the operator to enter the phrase.

In line 1030, if the length of the entry is greater than nineteen, the computer will return to line 1000 and allow the operator to enter another phrase.

Line 2000 clears the screen to prepare for the printing of the phrase.

Line 2010 establishes a FOR-NEXT loop with a beginning value of one and an ending value of the length of W$.

Line 2020 uses LEFT$( ) to extract I characters from the left of the phrase in the variable W2$.

Line 2030 uses RIGHT$( ) to extract I characters from the right of the phrase in the variable W3$.

Line 2040 prints the value of W2$ at TAB position one. Notice that a semicolon follows the print statement. The semicolon holds the print cursor at the space immediately following the printed data.

Line 2050 prints the right side of the phrase. To locate the TAB position, the program subtracts the length of W3$ from 40, the screen width. After TABbing to the calculated position, the second half of the phrase is printed.

Line 2060 completes the print loop.

Line 3000 positions the cursor near the bottom of the screen at line 21.

Line 3010 allows the operator to either enter an "N" to end the program or a "Y" to enter another word.

If a "Y" was entered, line 3020 sends the computer to line 1000 to allow another entry.

If the answer is not "N", an invalid entry was made and line 3030 sends the computer back to line 3000.

Line 9000 clears the screen.

Line 9010 ends the program.

LESSON TEN, #2

```
10  REM : LESSON 10 ADDITIONAL 2
20  REM  SAVE LES10A2
50  DIM L$(26),NL$(26)
500  DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
510  DATA Z,Y,X,W,V,U,T,S,R,Q,P,O,N,M,L,K,J,I,H,G,F,E,D,C,B,A
600  FOR I = 1 TO 26
610  READ L$(I)
620  NEXT I
700  FOR I = 1 TO 26
710  READ NL$(I)
720  NEXT I
997  REM  ::::::::::::
998  REM  :: GIVE OPTIONS
999  REM  ::::::::::::
1000  HOME
1010  PRINT  TAB( 09);"MESSAGE ENCODER & DECODER"
1020  VTAB 8: HTAB 10
1030  PRINT "1 - ENCODE MESSAGE"
1040  VTAB 10: HTAB 10
1050  PRINT "2 - DECODE MESSAGE"
1060  VTAB 20
1070  INPUT "ENTER SELECTION (0=END) ";O
1080  IF O = 0 THEN 9000
1090  ON O GOTO 2000,3000
1100  GOTO 1000
1997  REM  ::::::::::::::::
1998  REM  :: ENCODE MESSAGE
1999  REM  ::::::::::::::::
2000 W$ = "ENCODED"
2010  GOSUB 7000
2020  FOR I = 1 TO  LEN (S$)
2030 P$ =  MID$ (S$,I,1)
2040  FOR J = 1 TO 26
2050  IF P$ = L$(J) THEN P$ = NL$(J): GOTO 2070
2060  NEXT J
2070 S2$ = S2$ + P$
2080  NEXT I
2090  GOSUB 4000
2100  GOTO 1000
2997  REM  ::::::::::::::
```

```
2998  REM  :: DECODE MESSAGE
2999  REM  ::::::::::::::
3000 W$ = "DECODED"
3010  GOSUB 7000
3020  FOR I = 1 TO  LEN (S$)
3030 P$ =  MID$ (S$,I,1)
3040  FOR J = 1 TO 26
3050  IF P$ = NL$(J) THEN P$ = L$(J): GOTO 3070
3060  NEXT J
3070 S2$ = S2$ + P$
3080  NEXT I
3090  GOSUB 4000
3100  GOTO 1000
3997  REM  ::::::::::::::
3998  REM  :: PRINT RESULTS
3999  REM  ::::::::::::::
4000  HOME : VTAB 6: HTAB 11
4010  PRINT "THE OLD MESSAGE WAS"
4020  VTAB 8: HTAB 1
4030  PRINT S$
4040  VTAB 12: HTAB 11
4050  PRINT "THE NEW MESSAGE IS"
4060  VTAB 14: HTAB 1
4070  HTAB 1
4080  PRINT S2$
4090  VTAB 20
4100  INPUT "PRESS (RETURN) TO CONTINUE ";N$
4110  RETURN
6997  REM  ::::::::::::::
6998  REM  :: GET MESSAGE
6999  REM  ::::::::::::::
7000  HOME
7010 S$ = ""
7020 S2$ = ""
7030  PRINT "ENTER MESSAGE TO BE ";W$
7040  INPUT S$
7050  RETURN
8997  REM  ::::::::::::
8998  REM  :: END PROGRAM
8999  REM  ::::::::::::
9000  HOME
9010  END
```

Lines 10 and 20 are the opening remarks.

Line 50 dimensions the arrays to be used in the program; the L$ array to hold the standard characters, and the NL$ array to hold the code characters.

Line 500 defines the standard characters (A-Z).

Line 510 defines the code characters (Z-A). The example solution simply defines the code as the standard characters in reverse. Any combination of the 26 characters would work.

Lines 600 through 620 READ the first set of DATA elements, the standard characters, into the L$ array.

Lines 700 through 720 READ the second set of DATA elements, the code characters, into the NL$ array.

Line 1000 clears the screen.

Lines 1010 through 1070 print the options available to the operator on the CRT: "1 - ENCODE, 2 - DECODE or 0 TO END".

Line 1060 uses VTAB to position the cursor to screen line 20.

Line 1070 prompts the operator to enter either a zero to end the program or the number of the desired option and allows the option to be entered. The option selected is stored in O.

The operator's entry is checked in line 1080. If zero was entered, the program jumps to 9000, the ending section.

The ON-GOTO command in line 1090 sends the computer to the proper section of the program depending on the option that was entered.

Line 1100 sends the computer back to line 1000 if an invalid option was selected.

Line 2000 is the beginning of the encoding section. It assigns the word "ENCODED" to the variable W$.

Line 2010 executes the subroutine beginning at line 7000. This subroutine allows the operator to enter the phrase to be encoded or decoded and stores the phrase in S$.

In line 2020 the encoding loop is defined to repeat one time for each character in the variable S$, which is the string to be encoded.

Line 2030 assigns the character to be replaced to the variable P$.

The loop in line 2040 is used to search the array L$ which contains the standard characters.

If a match is found in line 2050, the character contained in the "J" element of the L$ array is replaced with the corresponding character in the code character array, NL$. Then the program jumps out of the loop to line 2070.

Line 2060 completes the J loop.

The coded version of the phrase is built in line 2070 by concatenating the code letter to the string S2$ which stores the encoded version of the message.

Line 2080 completes the encoding loop.

The GOSUB 4000 command in line 2090 executes the subroutine which prints the standard and coded messages for the operator to see.

Line 2100 sends the computer back to the beginning of the main section of the program for another entry.

Lines 3000 through 3100 are the decoding routine.

The process is exactly the same as the encoding section except that line 3050 uses NL$ as the array to be searched and L$ as the array used to build the decoded version of the message.

Lines 4000 through 4100 print the standard and coded messages for the operator to view and uses a prompted INPUT to stop the program.

Line 4110 is the RETURN command that terminates the subroutine.

Lines 7000 through 7050 are the entry routine.

Lines 7010 and 7020 initialize the variables that are used to hold the standard and coded messages. To initialize the string variables, they are assigned a "NULL" value (made equal to nothing).

Line 7030 prompts the operator to enter the message to be encoded or decoded. W$ is defined in the encoding and decoding sections of the program.

Line 7040 allows the operator to enter the phrase and stores the phrase in S$.

Line 7050 ends the subroutine.

Line 9000 clears the screen.

Line 9010 ends the program.

LESSON ELEVEN, #1

```
10  REM : LESSON 11 ADDITIONAL 1
20  REM  SAVE LES11A1
500 OI = 9:OB = 4
1000  HOME
1010  GR
1020  COLOR= OI
2000  VLIN 18,22 AT 19
2010  HLIN 18,20 AT 19
2020  HLIN 17,21 AT 20
2030  HLIN 18,20 AT 21
3000  COLOR= OB
4000  PLOT 19,16
4010  HLIN 18,20 AT 17
4020  HLIN 17,18 AT 18
4030  HLIN 16,17 AT 19
4040  HLIN 15,16 AT 20
4050  HLIN 16,17 AT 21
4060  HLIN 17,18 AT 22
4070  HLIN 18,20 AT 23
4080  HLIN 20,21 AT 18
4090  HLIN 21,22 AT 19
4100  HLIN 22,23 AT 20
4110  HLIN 21,22 AT 21
4120  HLIN 20,21 AT 22
4130  PLOT 19,24
5000  VTAB 23: PRINT "(0'S = END)"
5010  VTAB 22: INPUT "ENTER INNER COLOR,BORDER COLOR ";NI,NB
5020  IF NI = 0 OR NB = 0 THEN 9000
5030  FOR I = 15 TO 24
5040  FOR J = 15 TO 24
5050  IF  SCRN( I,J) = OI THEN  COLOR= NI: PLOT I,J
5060  IF  SCRN( I,J) = OB THEN  COLOR= NB: PLOT I,J
5070  NEXT J
5080  NEXT I
5090 OI = NI:OB = NB
5100  GOTO 5000
9000  TEXT : HOME : END
```

Lines 10 and 20 are the opening remarks.

Line 500 is a multiple statement line which assigns the original color numbers for the interior and border of the diamond.

OI, original interior, is assigned the color 9, orange. OB, original border is assigned the color 4, dark green.

Line 1000 clears the screen.

Line 1010 initiates Lo-Res graphics mode.

Line 1020 sets the color for the first drawing section.

Line 2000 draws a vertical line from position 18 to position 22 at column 19.

Line 2010 draws a horizontal line from position 18 to position 20 on line 19.

Line 2020 draws a horizontal line from position 17 to position 21 on line 20.

Line 2030 draws a horizontal line from position 18 to position 20 on line 21.

Line 3000 sets the color for the second drawing section.

Line 4000 plots the single point that is the top point of the diamond.

Line 4010 draws a horizontal line from position 18 to position 20 on line 17.

Line 4020 draws a horizontal line from position 17 to position 18 on line 18.

Line 4030 draws a horizontal line from position 16 to position 17 on line 19.

Line 4040 draws a horizontal line from position 15 to position 16 on line 20.

Line 4050 draws a horizontal line from position 16 to position 17 on line 21.

Line 4060 draws a horizontal line from position 17 to position 18 on line 22.

Line 4070 draws a horizontal line from position 18 to position 20 on line 23.

Line 4080 draws a horizontal line from position 20 to position 21 on line 18.

Line 4090 draws a horizontal line from position 21 to position 22 on line 19.

Line 4100 draws a horizontal line from position 22 to position 23 on line 20.

Line 4110 draws a horizontal line from position 21 to position 22 on line 21.

Line 4120 draws a horizontal line from position 20 to position 21 on line 22.

Line 4130 plots the bottom point of the diamond.

Line 5000 VTABs to the third line of the text window, line 23, and prints the message "(0'S = END)".

Line 5010 VTABs to the second line of the text window and uses a prompted INPUT to allow the operator to enter the new border and inner colors for the display. A multiple INPUT is used to store the operator's entries in NI and NB.

Line 5020 sends the computer to the ending section of the program if a zero was entered in line 5010 for either the new inner color or the new border color.

Line 5030 and 5040 begin the double nested color change loops.

The loops begin at 15 and end at 24 because the lowest line and column of our drawing is 15 and the highest line and column is 24. Reducing the size of the search area increases the execution speed of this section since fewer screen locations must be tested.

Line 5050 checks for the original inner color. If the color is found, the new inner color is used to PLOT that position.

Line 5060 checks for the original border color. If the color is found, the new border color is used to PLOT that position.

Line 5070 ends the J loop.

Line 5080 ends the I loop.

Line 5090 assigns the new colors to the original color variables so the operator can change the colors again.

Line 5100 returns the program to line 5000 so the operator can change the colors again.

Line 9000 returns the computer to TEXT mode, clears the screen and ends the program.

LESSON ELEVEN, #2

```
10  REM : LESSON 11 ADDITIONAL 2
20  REM  SAVE LES11A2
1000  HOME
1010  GR
1020  FOR I = 1 TO 500
1030  FOR X = 19 TO 0 STEP  - 1
1040 C = C + 1
1050  IF C = 16 THEN C = 0
1060  COLOR= C
1070 Y = 39 - X
1080  HLIN X,Y AT X
1090  HLIN Y,X AT Y
1100  VLIN X,Y AT X
1110  VLIN Y,X AT Y
1120  NEXT X
1130  NEXT I
9000  TEXT
9010  HOME
9020  END
```

Lines 10 and 20 are the opening remarks.

Line 1000 clears the screen.

Line 1010 turns on Lo-Res graphics mode.

Line 1020 defines the number of times the rectangles will be drawn; 500 times.

Line 1030 causes the rectangles to be drawn from the inside to the outside of the screen by stepping backwards from 19 to zero.

Line 1040 adds one to the variable C. C is used as the color for the drawing of the rectangle.

Line 1050 keeps the color in the range of 0 to 15 by assigning C a value of zero if C equals 16.

Line 1060 sets the color to the value of C.

Line 1070 subtracts the number X from 39. This calculation is used to determine the value of the Y coordinate. The X coordinate is used for the top and left sides of the rectangles and decreases in value because zero is the top row and the left-most column. The Y coordinate is used for the bottom and right sides of the rectangles and increases in value because 39 is the bottom row and the right-most column.

Line 1080 draws the top line of the rectangle.

Line 1090 draws the bottom line of the rectangle.

Line 1100 draws the left side of the rectangle.

Line 1110 draws the right side of the rectangle.

Line 1120 completes the X loop.

Line 1130 completes the I loop.

Line 9000 returns the computer to TEXT mode.

Line 9010 clears the screen.

Line 9020 ends the program.

LESSON TWELVE, #1

```
10  REM : LESSON 12 ADDITIONAL 1
20  REM  SAVE LES12A1
120 SB = 160:AR = 136:AL = 149
130 KB =  - 16384:KS =  - 16368
210 Y = 2:YP = 2:X = 20:D = 1
1000  HOME : GR
1010  IF D = 1 THEN Y = Y + 1: IF Y > 39 THEN Y = 0
1020  IF D = 2 THEN Y = Y - 1: IF Y < 0 THEN Y = 39
1030  COLOR= 2
1040  PLOT X,Y
1050  COLOR= 0
1060  PLOT X,YP
1070 YP = Y
1080 K =  PEEK (KB)
1090  IF K = AL THEN D = 2: POKE KS,0
1100  IF K = AR THEN D = 1: POKE KS,0
1110  IF K = SB THEN 9000
1120  GOTO 1010
9000  TEXT : HOME
9010  END
```

Lines 10 and 20 are the opening remarks.

Line 120 assigns the numeric codes for the keys. Space bar (SB) is 160, arrow-right (AR) is 136 and arrow-left (AL) is 149.

Line 130 assigns the appropriate addresses for the keyboard PEEK( ) and POKE.

Line 210 defines the beginning position for the drawing of the block of color and assigns the initial direction as one, down.

Line 1000 clears the screen and turns on graphics mode.

Line 1010 adds one to the Y, or row coordinate if the direction is one, down. Then the line makes the block reappear on the top if it goes off the bottom.

Line 1020 subtracts one from the Y, or row coordinate if the direction is two, up. Then the line makes the block reappear on the bottom if it goes off the top.

Line 1030 sets the color to two, blue.

Line 1040 plots the block at its new coordinates.

Line 1050 sets the color at zero, black. This is in preparation for erasing the block.

Line 1060 erases the block at its previous coordinates.

Line 1070 prepares for the next animation step by making the previous row coordinate equal to the current row coordinate.

Line 1080 PEEKs at the keyboard.

Line 1090 changes the direction to two, up, if the arrow-left key was pressed. Then the line resets the keyboard with the keyboard POKE command.

Line 1100 changes the direction to one, down, if the arrow-right key was pressed. The line then resets the keyboard with the keyboard POKE command.

Line 1110 sends the program to the ending section if the space bar was pressed.

Line 1120 sends the program to the top of the main section for the next animation step.

Line 9000 returns the computer to text mode and clears the screen.

Line 9010 ends the program.

LESSON TWELVE, #2

```
10  REM : LESSON 12 ADDITIONAL 2
20  REM SAVE LES12A2
40  REM :: HOUSEKEEPING SECTION
100 C = 15
120 IK = 201:MK = 205:JK = 202:KK = 203:SB = 160
130 KB =  - 16384:KS =  - 16368
140 SD =  - 16336
150  HOME : GR
190  REM :: MAKE STAR FIELD
200  FOR I = 1 TO 40
210 X =  INT (38 *  RND (1) + 1)
220 Y =  INT (38 *  RND (1) + 1)
230 SC =  INT (14 *  RND (1) + 1)
240  COLOR= SC
250  PLOT X,Y
260  NEXT I
270  VTAB 21: INPUT "PRESS RETURN TO BEGIN: ";A$
280  HOME
490  REM :: STARTING POSITION
500 X = 20:Y = 39
510 XP = X:YP = Y
520 B = 0
990  REM :: BEGIN ANIMATION
1000 D = 0: GOSUB 3000
1010 PK =  PEEK (KB)
1020  IF PK = IK THEN D = 1: POKE KS,0: GOSUB 3100
1030  IF PK = JK THEN D = 2: POKE KS,0: GOSUB 3100
1040  IF PK = MK THEN D = 3: POKE KS,0: GOSUB 3100
1050  IF PK = KK THEN D = 4: POKE KS,0: GOSUB 3100
1060  IF PK = SB THEN 9000
```

```
107Ø  IF D = Ø THEN 151Ø
108Ø  ON D GOTO 11ØØ,12ØØ,13ØØ,14ØØ
109Ø  REM :: DIRECTION 1=UP
11ØØ Y = Y - 1: IF Y < Ø THEN 4ØØØ
1110  GOTO 15ØØ
1190  REM :: DIRECTION 2=LEFT
12ØØ X = X - 1: IF X < Ø THEN X = 39
1210  GOTO 15ØØ
1290  REM :: DIRECTION 3=DOWN
13ØØ Y = Y + 1: IF Y > 39 THEN Y = 39
1310  GOTO 15ØØ
1390  REM :: DIRECTION 4=RIGHT
14ØØ X = X + 1: IF X > 39 THEN X = Ø
15ØØ HIT =  SCRN( X,Y): IF HIT <  > Ø THEN 5ØØØ
1510  COLOR= C: PLOT X,Y
1520  COLOR= B: PLOT XP,YP
1530 XP = X
1540 YP = Y
1550  FOR I = 1 TO 5
1560 SX =  INT (4Ø *  RND (1)):SY =  INT (4Ø *  RND (1))
1570 SC =  INT (14 *  RND (1) + 1): COLOR= SC
1580  PLOT SX,SY
1590  IF X = SX AND Y = SY THEN 5ØØØ
16ØØ  NEXT I
2ØØØ  GOTO 1ØØØ
2990  REM :: SOUND SUBROUTINE
3ØØØ  FOR I = 1 TO 1Ø
3Ø1Ø S =  PEEK (SD):S =  PEEK (SD)
3Ø2Ø  NEXT I
3Ø3Ø  RETURN
31ØØ  FOR I = 1 TO 1Ø
3110 S =  PEEK (SD)
3120  NEXT I
3130  RETURN
32ØØ S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =
     PEEK (SD)
3210 S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =
     PEEK (SD)
322Ø S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =
     PEEK (SD)
3230 S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =
     PEEK (SD)
3240  RETURN
33ØØ  FOR I = 1 TO 1Ø
3310 S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =  PEEK (SD):S =
     PEEK (SD)
3320  NEXT I
3330  RETURN
4ØØØ  FOR J = 1 TO 1Ø: GOSUB 32ØØ: NEXT J
4Ø1Ø  GOSUB 32ØØ: VTAB 21: PRINT "YOU WIN": GOTO 9ØØØ
5ØØØ  FOR J = 1 TO 1Ø: GOSUB 33ØØ: NEXT J
5Ø1Ø  GOSUB 33ØØ: VTAB 21: PRINT "GOT 'CHA"
9ØØØ  VTAB 23: INPUT "PRESS RETURN ";A$
9Ø1Ø  TEXT : HOME
9Ø2Ø  END
```

Much of the answer to the second additional assignment is the same as the "SPACE SHIP" program answer explained in the VideoSoft answer to the assignment for Lesson Twelve.

Lines 10, 20 and 40 are the opening remarks.

Line 100 assigns the color white, 15, to C.

The I, J, K and M keys are used to allow the operator to control the movement of the white block, and the space bar to end the animation.

Therefore, line 120 assigns the numeric codes to the five keys.

"IK", the "I" key is 201.

"MK", the "M" key is 205.

"JK", the "J" key is 202.

"KK", the "K" key is 203.

"SB", the space bar is 160.

In line 130 the keyboard PEEK and POKE addresses are assigned to KB and KS.

Line 140 assigns the sound's PEEK address to SD.

Line 150 clears the screen and switches to graphics mode.

The field of blocks is drawn by plotting 40 random blocks of color on the CRT.

The loop in line 200 simply controls the number of blocks to be plotted, 40 stars.

Line 210 uses the RND( ) and the INT( ) commands to pick a random number between one and 38.

The random number is assigned to X to be used as the horizontal, or column coordinate for each block.

Line 220 uses an identical random function to select a vertical, or row coordinate for each block.

Line 230 uses another random function to assign a color between 1 and 14 to the variable SC. It omits the colors black (zero), and white (15), because black is the dominant background color and white is the color of the block moved by the operator.

Line 240 sets the color to SC.

Line 250 plots a random block of color.

The NEXT I command in line 260 executes the next pass of the loop.

Line 270 uses a dummy INPUT to hold the initial field of blocks on the screen until the operator presses the "RETURN" key.

Line 280 will clear only the text window at the bottom of the screen.

Line 500 sets the starting coordinates at the middle, bottom of the screen, column 20, row 39.

Line 510 assigns the current coordinates to the previous coordinates.

Line 520 assigns the erasing color, black, to B. Since the operator must move the white block around the colors rather than over the colors, the erasing color will always be black.

The main program module begins in line 1000.

Lines 1000-1070 allow the operator to set the direction of movement based on the last key pressed by the operator.

Line 1000 initializes the value of D as zero at the beginning of each animation sequence and executes a sound subroutine.

Line 1010 PEEKs at the keyboard.

Lines 1020-1060 test the number from the keyboard to see if one of the direction keys was pressed or the space bar was pressed and execute a subroutine which creates a sound for each key pressed.

In line 1020 if the I key was pressed, D is given a value of one (up) and the keyboard is reset with "POKE KS,0". Then the sound subroutine at 3100 is executed.

Line 1030 assigns a value of 2, left, to the variable D if the J was pressed. Then line 1030 resets the keyboard and executes the sound subroutine in line 3100.

Line 1040 assigns a value of 3, down, to D if the M key was pressed, then resets the keyboard and executes the sound subroutine in line 3100.

Line 1050 assigns a value of 4, right, to D if the K key was pressed, resets the keyboard and executes the sound subroutine in line 3100.

In line 1060, if the space bar was pressed, the program goes to the ending routine in line 9000.

The IF-THEN in line 1070 instructs the computer to GOTO line 1510 if no direction has been assigned (D will equal zero).

Lines 1080-1400 were taken from the "ROAMING BLOCK" program explained in Lesson Twelve.

This section is the direction control section which moves the white block by adding or subtracting from the X or Y coordinates.

Only the up and down routines have been changed. The up routine has been changed to jump to the "Win" routine if the operator moves off the top of the screen.

The down routine has been changed to keep the operator on row 39 or above.

Lines 1500-1540 create the illusion of animation by PLOTing the white block and "erasing" by redrawing in black.

Line 1500 uses the SCRN( ) to test to see if the operator has moved the white block into a colored block. The number of the color is stored in the variable HIT. If the value of HIT is other than zero, black, the operator collided and the program jumps to the lose section in line 5000.

Line 1510 sets the color to C, 15, then PLOTs the white block at X,Y.

Line 1520 sets the color to B, the background color, and restores the background by PLOTting at XP, YP.

Lines 1530 and 1540 store the new coordinates, X,Y, in the previous coordinate variables, XP,YP.

Lines 1550 through 1600 PLOT 5 additional random blocks of color. Line 1590 jumps to the lose routine if one of the random colors is the same as the current position of the white block.

Line 2000 sends the computer to the beginning of the animation sequence for the next movement.

Lines 3000 through 3030 comprise the first sound subroutine executed on each step of the animation.

Line 3000 establishes a FOR-NEXT loop with 10 passes.

Line 3010 executes two sound PEEK commands on each loop pass.

Line 3020 ends the loop.

Line 3030 ends the subroutine with the command word RETURN.

Lines 3100-3130 are the second sound routine executed when the white block moves.

These lines are the same as the first subroutine except in line 3110 only one sound routine is executed on each of the ten loop passes.

Lines 3200-3230 are the third sound routine which executes if the operator wins.

Line 3200 is a multiple statement line which executes the sound PEEK command five times.

Lines 3210, 3220 and 3230 are identical to line 3200.

The subroutine ends with the RETURN command in line 3240.

Lines 3300 through 3320 are the fourth sound subroutine which executes if the operator loses.

Line 3300 establishes a loop with 10 passes.

Line 3310 executes five sound PEEK commands.

The loop ends with the NEXT I in line 3320.

The subroutine ends with the RETURN command.

All four of the subroutines look similar, yet each produces a different sound.

Lines 4000 and 4010 are the win section.

Line 4000 executes the 3200 subroutine ten times.

Line 4010 executes the 3200 subroutine one more time, VTABs to line 21, PRINTs "YOU WIN" and jumps to line 9000.

Lines 5000 and 5010 are the lose section.

Line 5000 executes the 3300 subroutine ten times.

Line 5010 executes the 3300 subroutine one more time, VTABs to line 21 and PRINTs "GOT 'CHA".

Line 9000 ends the program by VTABbing to line 23, the third line of the text window, and then uses a prompted INPUT to hold the screen until the operator presses the "RETURN" key.

Line 9010 returns the computer to TEXT mode and clears the screen.

Line 9020 ENDs the program.

LESSON THIRTEEN, #1

```
10  REM : LESSON 13 ADDITIONAL 1
20  REM  SAVE LES13A1
50  DIM C$(10,3)
1000  HOME
```

```
1010  PRINT  TAB( 10);"<<< SCHOOL SCHEDULE >>>"
1020  VTAB 4: PRINT  TAB( 10);"1 = ENTER NEW SCHOOL SCHEDULE"
1030  VTAB 5: PRINT  TAB( 10);"2 = CHANGE SCHOOL SCHEDULE"
1040  VTAB 6: PRINT  TAB( 10);"3 = VIEW SCHOOL SCHEDULE"
1050  VTAB 8: INPUT "      ENTER NUMBER OF CHOICE (0=END) ";A
1060  IF A = 0 THEN 9000
1070  ON A GOTO 2000,3000,4000
1080  GOTO 1000
1997  REM  ::::::::::::::
1998  REM  :: ENTER NEW SCHEDULE
1999  REM  ::::::::::::::
2000  FOR I = 1 TO 10
2010  HOME
2020  VTAB 4: HTAB 5: PRINT "CLASS #";I
2030  VTAB 7: INPUT "ENTER CLASS NAME ";C$(I,1)
2040  VTAB 8: INPUT "ENTER STARTING TIME ";C$(I,2)
2050  VTAB 9: INPUT "ENTER ENDING TIME ";C$(I,3)
2060  NEXT I
2070  GOSUB 7000
2080  GOTO 1000
2997  REM  ::::::::::::::
2998  REM  :: CHANGE SCHEDULE
2999  REM  ::::::::::::::
3000  GOSUB 8000
3010  GOSUB 6000
3020  VTAB 18: HTAB 1
3030  INPUT "ENTER NUMBER TO CHANGE (0=END) ";N
3040  IF N = 0 THEN 3160
3050  IF N < 1 OR N > 10 THEN 3030
3060  VTAB 18: PRINT  TAB( 40);
3070  HTAB 1: PRINT "    CLASS #";N
3080  VTAB 19: INPUT "ENTER CLASS NAME ";C$(N,1)
3090  VTAB 20: INPUT "ENTER BEGINNING TIME ";C$(N,2)
3100  VTAB 21: INPUT "ENTER ENDING TIME ";C$(N,3)
3110  VTAB 18: PRINT  TAB( 40);: HTAB 1
3120  VTAB 19: PRINT  TAB( 40);: HTAB 1
3130  VTAB 20: PRINT  TAB( 40);: HTAB 1
3140  VTAB 21: PRINT  TAB( 40);: HTAB 1
3150  GOTO 3010
3160  GOSUB 7000
3170  GOTO 1000
3997  REM  ::::::::::::::::
3998  REM  :: VIEW DATA
3999  REM  ::::::::::::::::
4000  GOSUB 8000
4010  GOSUB 6000
4020  VTAB 19: HTAB 1: INPUT "PRESS RETURN TO CONTINUE ";N$
4030  GOTO 1000
5997  REM  ::::::::::::::
5998  REM  :: PRINT DATA ON CRT
5999  REM  ::::::::::::::
6000  HOME
6010  PRINT  TAB( 10);"<<<< SCHEDULE DATA >>>>"
6020  FOR I = 1 TO 10
6030  VTAB I + 3
6040  HTAB 1: PRINT "#";I;
6050  HTAB 5: PRINT C$(I,1);
6060  HTAB 26: PRINT C$(I,2);
6070  HTAB 34: PRINT C$(I,3);
```

Line 6030 positions the cursor at the beginning of the print line.

Line 6040 prints the class number starting at screen position 1.

Line 6050 prints the class name starting at screen position 5.

Line 6060 prints the beginning time starting at screen position 26.

Line 6070 prints the ending time starting at screen position 34.

Line 6080 ends the print loop.

Line 6090 ends the data printing subroutine.

Line 7000 prepares the file for use.

Line 7010 prepares the file to accept data.

Line 7020 starts the loop to print data to the file.

Line 7030 prints the class names to the file.

Line 7040 prints the beginning times to the file.

Line 7050 prints the ending times to the file.

Line 7060 completes the data print loop.

Line 7070 closes the file.

Line 7080 ends the subroutine.

Line 8000 prepares the file for use.

Line 8010 prepares the file to retrieve data.

Line 8020 starts the data input loop.

Line 8030 inputs the class names from the file.

Line 8040 inputs the beginning times from the file.

Line 8050 inputs the ending times from the file.

Line 8060 ends the data input loop.

Line 8070 closes the file.

Line 8080 ends the subroutine.

Line 9000 when executed will clear the screen and end the program.

LESSON THIRTEEN, #2

```
1Ø  REM : LESSON 13 ADDITIONAL 2
2Ø  REM  SAVE LES13A2
1ØØØ  HOME
1Ø1Ø  PRINT  TAB( 8);"<< PROGRAM DESCRIPTIONS >>"
1Ø2Ø  VTAB 4: PRINT "     1 - ENTER PROGRAM DESCRIPTIONS"
```

```
1030  VTAB 5: PRINT "     2 - CHANGE PROGRAM DESCRIPTIONS"
1040  VTAB 6: PRINT "     3 - VIEW PROGRAM DESCRIPTIONS"
1050  VTAB 8: INPUT "     ENTER NUMBER OF CHOICE (0=END) ";A
1060  IF A = 0 THEN 9000
1070  ON A GOTO 2000,3000;4000
1080  GOTO 1000
1997  REM  ::::::::::::::
1998  REM  :: ENTRY SECTION
1999  REM  ::::::::::::::
2000  HOME
2010  GOSUB 7500
2020 FD = FD + 1
2030 D = FD
2040  VTAB 9: PRINT "ENTER PROGRAM NAME #";FD
2050  VTAB 10: INPUT "(PRESS RETURN TO END): ";D1$
2060  IF D1$ = "" THEN 1000
2070  VTAB 12: INPUT "ENTER PROGRAM DESCRIPTION: ";D2$
2080  GOSUB 8000
2090  GOSUB 8500
2100  GOTO 2020
2997  REM  ::::::::::::::
2998  REM  :: UPDATE SECTION
2999  REM  ::::::::::::::
3000  GOSUB 7500
3010  HOME
3020  VTAB 9: PRINT "ENTER PROGRAM NUMBER TO CHANGE"
3030  VTAB 10: INPUT "(ENTER 0 TO END) ";D
3040  IF D = 0 THEN 1000
3050  IF D < 0 OR D > FD THEN 3020
3060  GOSUB 7000
3070  HOME
3080  VTAB 7: PRINT "PROGRAM NUMBER ";D
3090  VTAB 9: PRINT "PROGRAM NAME: ";D1$
3100  VTAB 11: PRINT "PROGRAM DESCRIPTION: ";D2$
3110  VTAB 9: INPUT "PROGRAM NAME: ";D1$
3120  VTAB 11: INPUT "PROGRAM DESCRIPTION: ";D2$
3130  GOSUB 8000
3140  GOTO 3010
3997  REM  ::::::::::::
3998  REM  :: VIEW DATA
3999  REM  ::::::::::::
4000  HOME
4010  GOSUB 7500
4020  VTAB 10: PRINT "ENTER BEGINNING PROGRAM NUMBER"
4030  VTAB 12: INPUT "FOR PRINT (0=END) ";D
4040  IF D < 0 OR D > FD THEN 4020
4050  IF D = 0 THEN 1000
4060 TL = 0
4070  HOME
4080  GOSUB 7000
4090 TL = TL + 3 +  INT (( LEN (D2$) + 20) / 40) + 1
4100  IF TL > 21 THEN 4180
4110  PRINT "PROGRAM # ";D;" : ";D1$
4120  PRINT
4130  PRINT "DESCRIPTION ";D2$
4140  PRINT
4150 D = D + 1
4160  IF D = FD + 1 THEN  VTAB 22: INPUT "END OF FILE -
      PRESS (RETURN)";N$: GOT
```

```
4170  GOTO 4080
4180  VTAB 22: PRINT "PRESS (RETURN) TO CONTINUE"
4190  VTAB 23: INPUT "ENTER A LETTER TO END ";N$
4200  IF N$ <  > "" THEN 1000
4210 TL = 0
4220  HOME
4230  GOTO 4090
6997  REM  ::::::::::::::::
6998  REM  :: READ DESCRIPTION
6999  REM  ::::::::::::::::
7000  PRINT  CHR$ (4);"OPEN DESCRIPTIONS,L150"
7010  PRINT  CHR$ (4);"READ DESCRIPTIONS,R";D
7020  INPUT D1$
7030  INPUT D2$
7040  PRINT  CHR$ (4);"CLOSE DESCRIPTIONS"
7050  RETURN
7497  REM  ::::::::::::::::
7498  REM  :: READ ZERO RECORD
7499  REM  ::::::::::::::::
7500  PRINT  CHR$ (4);"OPEN DESCRIPTIONS,L150"
7510  PRINT  CHR$ (4);"READ DESCRIPTIONS,R0"
7520  INPUT FD
7530  PRINT  CHR$ (4);"CLOSE DESCRIPTIONS"
7540  RETURN
7997  REM  ::::::::::::::::
7998  REM  :: WRITE DATA
7999  REM  ::::::::::::::::
8000  PRINT  CHR$ (4);"OPEN DESCRIPTIONS,L150"
8010  PRINT  CHR$ (4);"WRITE DESCRIPTIONS,R";D
8020  PRINT D1$
8030  PRINT D2$
8040  PRINT  CHR$ (4);"CLOSE DESCRIPTIONS"
8050  RETURN
8497  REM  ::::::::::::::::
8498  REM  :: WRITE ZERO RECORD
8499  REM  ::::::::::::::::
8500  PRINT  CHR$ (4);"OPEN DESCRIPTIONS,L150"
8510  PRINT  CHR$ (4);"WRITE DESCRIPTIONS,R0"
8520  PRINT FD
8530  PRINT  CHR$ (4);"CLOSE DESCRIPTIONS"
8540  RETURN
9000  HOME : END
```

Lines 10 and 20 are the opening remarks.

Line 1000 clears the screen.

Line 1010 displays the program heading.

Line 1020 displays the first option available to the operator.

Line 1030 displays the second option available to the operator.

Line 1040 displays the third option available to the operator.

Line 1050 allows the operator to make a selection.

In Line 1060, if zero is entered, the program jumps to the ending section at line 9000.

Line 1070 sends the computer to the selected section of the program.

Line 1080 sends the program to the beginning of the menu if an invalid option was entered.

Line 2000 clears the screen.

Line 2010 executes the "READ ZERO RECORD" subroutine.

Line 2020 adds one to the variable FD. This variable is used to store the number of the last record written.

Line 2030 sets the variable D equal to the variable FD. D is used as the record number in the read and write subroutines.

Line 2040 prints a prompt to the operator and the number of the record being added.

Line 2050 displays the second part of the operator prompt and allows the operator to enter the program name.

Line 2060 sends the computer to the top of the program if no program name was entered.

Line 2070 allows the operator to enter a description of the program.

Line 2080 executes the "WRITE DATA" subroutine and stores the entered information in record number D.

Line 2090 executes the "WRITE ZERO RECORD" subroutine to update the last record counter.

Line 2100 returns the computer to line 2020 so the operator can continue making entries.

Line 3000 executes the "READ ZERO RECORD" subroutine.

Line 3010 clears the screen.

Line 3020 prints a prompt to the operator.

Line 3030 allows the operator to enter the number of a record containing program data he or she wishes to change.

Line 3040 sends the computer to the top of the program if no record number was entered.

Line 3050 checks the operator's entry. A valid record number cannot be less than zero or greater than the last record in the file.

Line 3060 executes the "READ DESCRIPTION" subroutine to retrieve the data to be changed.

Line 3070 clears the screen.

Line 3080 prints the record number whose data is to be edited.

Line 3090 prints the existing value of the program name.

Line 3100 prints the existing value of the program description.

Line 3110 positions the cursor to the location of the printed name and allows the operator to enter the new value.

Line 3120 positions the cursor to the location of the printed description and allows the operator to enter the new value.

Line 3130 executes the "WRITE DATA" subroutine.

Line 3140 returns the computer to the beginning of the update section.

Line 4000 clears the screen.

Line 4010 executes the "READ ZERO RECORD" subroutine.

Line 4020 prints part of an operator prompt.

Line 4030 allows the operator to enter a beginning program number for the data display.

Line 4040 error checks the operator's entry.

Line 4050 returns the computer to the top of the program if a zero was entered.

Line 4060 initializes the variable TL. TL is used to track the total lines printed on the screen.

Line 4070 clears the screen.

Line 4080 executes the "READ DESCRIPTION" subroutine.

Line 4090 calculates the number of lines that are used to print the data and adds them to the total lines printed.

Line 4100 sends the computer to line 4180 if the data from the last record read exceeds the number of lines available for printing.

Line 4110 prints the program number.

Line 4120 prints a blank line.

Line 4130 prints a program description.

Line 4140 prints a blank line.

Line 4150 adds one to D, advancing the record number to move forward through the file.

Line 4160 prints a message to the operator indicating that the last record has been printed and returns the computer to the top of the program if the last record is printed.

Line 4170 returns the program to line 4080 for another record.

Line 4180 prints a message to the operator.

Line 4190 allows the operator to enter a letter to end the program or press the "RETURN" key to continue.

Line 4200 sends the computer to line 1000 if the operator pressed the "RETURN" key.

Line 4210 sets the total lines printed to zero to prepare for the next screen of data.

Line 4220 clears the screen.

Line 4230 returns the computer to line 4090 to continue printing programs and descriptions.

Line 7000 opens the file with a record length of 150.

Line 7010 reads record number D.

Line 7020 inputs the program name from the file.

Line 7030 inputs the program description from the file.

Line 7040 closes the file.

Line 7050 ends the subroutine.

Line 7500 opens the file.

Line 7510 reads record number zero.

Line 7520 inputs the last record written counter.

Line 7530 closes the file.

Line 7540 ends the subroutine.

Line 8000 opens the file.

Line 8010 prepares the file to WRITE data.

Line 8020 prints the program name to the file.

Line 8030 prints the program description to the file.

Line 8040 closes the file.

Line 8050 ends the subroutine.

Line 8500 opens the file.

Line 8510 prepares to WRITE data to record number zero.

Line 8520 prints the last record written counter to the file.

Line 8530 closes the file.

Line 8540 ends the subroutine.

Line 9000 ends the program.

Evaluation Test #1
Lessons One And Two

DIRECTIONS — This test contains two parts. Each part is explained separately. Total points: 100.

PART ONE DIRECTIONS — Fill in the blank with the correct command, term or name. Each correct answer is worth two points.

1. ———————— A DOS command which instructs the computer to remove protection of the LOCK command.
2. ———————— The flashing block of light visible on the CRT.
3. ———————— A high level computer language used in this course to write programs.
4. ———————— Sets the computer's screen to print black letters on a white background.
5. ———————— Command that clears the computer's program memory.
6. ———————— Known as the "Father of the Computer".
7. ———————— A DOS command which instructs the computer to display a list of the files on diskette upon the CRT.
8. ———————— An error in a program.
9. ———————— Instructs the computer to ignore the statement line. Used for explanatory remarks of the programmer.
10. ———————— The electronic chip within the computer that interprets and executes instructions.
11. ———————— Where information is permanently stored until it is erased.
12. ———————— A high level language used for controlling the diskette and diskette operations.
13. ———————— If you had changed the computer's print speed to 64, this command would return the print speed to 255.
14. ———————— A set of instructions that the computer will execute.
15. ———————— A DOS command which instructs the computer to make a copy of the program file from a diskette and to place it in the memory of the machine.
16. ———————— The computer's display screen.
17. ———————— Displays the statement lines of the program currently stored in the computer's memory.
18. ———————— Process by which a diskette is marked and organized into tracks and sectors.
19. ———————— Instructs the computer to cease execution of program statements.
20. ———————— This key is like a shift key. It has to be used with another key.

21. ———————————— DOS command which, when followed by a program name, initializes a diskette by marking tracks and sectors.

22. ———————————— DOS command which instructs the computer to permanently erase a program from the diskette.

23. ———————————— Controls the print rate of the CRT.

24. ———————————— DOS command which, in conjunction with a program name, will first load the program from diskette and then cause it to be executed.

25. ———————————— A program that will RUN each time that the Disk Operating System (DOS) is booted.

26. ———————————— Clears the text screen and moves the cursor to the upper left-hand corner of the CRT.

27. ———————————— Sets the computer's screen to alternate from INVERSE to NORMAL PRINT.

28. ———————————— A DOS command which instructs the computer to change the name of a diskette file.

29. ———————————— Instructs the computer to display a value of a variable or a literal on the CRT.

30. ———————————— A DOS command which instructs the computer to prevent a diskette file from being erased or altered.

31. ———————————— A numbered set of computer instructions and/or commands within a program.

32. ———————————— A command that executes immediately without the need of a program.

33. ———————————— A DOS command which instructs the computer to make a copy of the program currently in memory within the computer and store it on a diskette.

34. ———————————— The slowest print rate of the CRT.

35. ———————————— Refers to any letter, digit, punctuation mark and the space.

PART TWO DIRECTIONS — Write the answers in the space provided. Please note that questions 2, 3, and 4 are combined into one question. Each correct answer is worth three points.

1. The syntax for the SPEED command is:

2—4. The three rules for naming files are:

5. The structure of the RENAME command is:

6. How can you tell if a file is protected on your diskette?

7. To list a certain range of line numbers, you type:

8. If you wished to print a message on the screen that says, COMPUTER PROGRAMMING IS FUN, you would type:

9. If you wished to place twenty **A**'s on the CRT and do it as quickly as possible, which two keys you would use?

10. If the PRINT command is written in a statement line by itself, what occurs?

Evaluation Test #2
Lessons Three And Four

Directions — This test contains three parts. Each part is explained separately. Total points: 100.

PART ONE DIRECTIONS— Fill in the blank with the correct word, command, or symbol. Each correct answer is worth 1 point.

1. ———————— Prints a message prior to executing an INPUT command which requires an operator response.
2. ———————— Prints a sufficient number of blanks to move the cursor from its current position to the position specified by the number within the parentheses.
3. ———————— Instructs the computer to perform exponentiation (perform repeated multiplication).
4. ———————— When assigning a value to a character variable, the value must be enclosed within these.
5. ———————— Used to directly assign values to variables (usually assumed).
6. ———————— When used in a PRINT statement, moves the cursor to the beginning of the next print zone.
7. ———————— Character variable names end with this symbol.
8. ———————— Instructs the computer to add.
9. ———————— A place in the memory of the computer in which information is stored.
10. ———————— Places the cursor at a specific position on the 255 character print line. (Does **not** print blanks to position the cursor.)
11. ———————— Instructs the computer to divide.
12. ———————— A series of adjacent letters, numbers, symbols or spaces enclosed within quotation marks.
13. ———————— Allows the operator to enter a value from the keyboard which will be assigned to a variable.
14. ———————— Cannot contain blanks, reserved words, or punctuation marks.
15. ———————— A message on the screen that tells the operator that information is being requested and should be typed on the keyboard.
16. ———————— Positions the cursor to any one of the twenty-four text screen lines.
17. ———————— Instructs the computer to multiply.
18. ———————— When used within a PRINT statement, moves the cursor to the next space on the print line.
19. ———————— Instructs the computer to subtract.
20. ———————— A label used to identify a particular variable which must begin with a letter of the alphabet.

PART TWO DIRECTIONS — Choose the BEST answer and place the letter of your choice in the space provided. Each correct answer is worth 5 points.

——— 1. A prompted input has three parts written in this order:

A. prompted INPUT, command, variable.
B. command, prompt and variable name.
C. variable, command, prompt.
D. prompt, variable name, command.

——— 2. The line:

1000 INPUT "ENTER NAME ";N$

will

A. PRINT a name on the screen.
B. assign a name to the variable "INPUT".
C. instruct the operator to enter a name.
D. PRINT the name stored in the INPUT command.

——— 3. A number can be included in a character variable name as long as

A. the name begins with a letter and ends with a dollar sign.
B. it is at least two digits in length.
C. the number is written after the dollar sign.
D. the number is enclosed within quotation marks.

——— 4. The line:

1000 PRINT A$

instructs the computer to

A. PRINT the characters A and $ on the screen for the operator.
B. store the information in variable A$.
C. assign new information to the variable A$.
D. PRINT the information stored in variable A$.

——— 5. The commands:

```
A = 4548/6
PRINT A
```

will instruct the computer to

A. divide 4548 by 6 and PRINT the answer 758.
B. PRINT "A=4548/6=758"
C. multiply 4548 times 6 and PRINT the answer 27288.
D. prompt the operator to enter the answer to 4548/6.

——— 6. In the calculation:

A-(4+5) * 7^5 - 342/2

the calculation that will be performed first will be

A. 7^5 B. 342/2 C. 4+5 D. 7^5-342

——— 7. In line:

1000 PRINT,,"Age"

the two commas after the PRINT command instruct the computer to

A. leave the next two lines blank and print "Age" on the third line.
B. skip zones one and two and print "Age" in zone three.
C. PRINT two commas in the PRINT statement line.
D. PRINT "Age" after two commas are entered.

——— 8. In line:

1010 PRINT "THE SCORE ";S$;" IS EXCELLENT"

we have

A. instructed the computer to locate the score that is excellent.
B. made an error and put in one too many semicolons.
C. used semicolons to make the line easier to read.
D. used semicolons to PRINT a variable and string literals on the same line.

PART THREE DIRECTIONS — For all questions in this part you are to explain what is happening in the lines selected from the short program which follows. Each correct answer is worth 8 points.

```
 500 SPEED = 0
 510 INVERSE
1000 HOME
1020 HTAB 20
1030 PRINT "1ST HTAB";
1040 HTAB 1
1050 PRINT "2ND HTAB"
1060 VTAB 4
1070 PRINT TAB (5);"1ST TAB"; TAB (25);"2ND TAB"
1080 SPEED =255
1090 NORMAL
1100 END
```

EXPLAIN WHAT IS HAPPENING IN EACH OF THESE LINES:

LINE 500

LINE 1020

LINE 1060

LINE 1070

LINE 1090

Evaluation Test #3
Lessons Five And Six

DIRECTIONS — This test is divided into three parts. Each section is explained separately. Total points: 100.

PART ONE DIRECTIONS — Write the correct answer in the space provided. Each correct answer is worth 2 points.

1. ———————— A set of programming lines that are repeated over and over until CTRL-C or CTRL-RESET stops the program.

2. ———————— Used to make the computer "bell" sound.

3. ———————— Symbol used by the computer to compare; meaning "less than".

4. ———————— Establishes two or more conditions to be met in the "IF" part of an IF-THEN statement. ("IF" all conditions are met "THEN".)

5. ———————— Instructs the computer to jump lines within a program to a specific line number.

6. ———————— Something that is good programming practice before you sit down to enter your programs at the keyboard.

7. ———————— Temporarily stops the listing of a program until another key is pressed.

8. ———————— First part of this command sets up a condition to be met. Should the condition be true, the computer will execute all instructions following the second part of the command. Should the condition be false, the computer goes to the next numbered statement line.

9. ———————— While comparing two conditions, the computer reads this symbol as meaning "not equal to".

10. ———————— An INPUT statement used to temporarily stop the program until the operator is ready to proceed.

11. ———————— The command used to remove specific lines from a program.

12. ———————— Establishes two or more conditions in the "IF" part of the IF-THEN statement. "IF" either condition is met, the "THEN" portion will be executed.

13. ———————— Symbol used by the computer meaning "greater than".

14. ———————— This can be substituted for the PRINT command.

15. ———————— Can be used to interrupt the running or listing of a program. When used, will display the line number the computer was executing when interrupted.

16. ———————— When using more than one input variable in a single INPUT statement, the variables must be separated by these.

17. ———————— An INPUT statement using more than one variable allowing the operator to enter more than one item.

18. ———————————— Can be used to interrupt the running or listing of a program. Does not display line number the computer was executing when interrupted.

19. ———————————— To place more than one command on a line you use this to separate the commands.

20. ———————————— Symbol used by computer in comparisons meaning "equal".

DIRECTIONS FOR PART TWO — Explain what is happening in each of these program lines. A space has been provided for your answer. Each correct answer is worth three points.

1. 1610 IF N<1 OR N>5 THEN PRINT "ENTER ANOTHER NUMBER"

2. 1020 IF N<>0 GOTO 1000

3. 1310 INPUT "WAS YOUR SCORE HIGHER THAN 85% ? (YES OR NO) ";AN$

4. 1200 PRINT "THE SCORE YOU ENTERED WAS ";S

5. 2000 INPUT "(PRESS RETURN TO CONTINUE) ";A$

6. 1040 PRINT CHR$(7)

7. 1480 BL$ = CHR$(7) : PRINT BL$

8. 1030 IF G = 0 THEN 4000

9. 1010 HOME: PRINT BL$: VTAB 8 : FLASH

10. 10 REM ::ASSIGNMENT #2

PART THREE DIRECTIONS — Carefully read the following program lines. If an error occurs in a program line, rewrite the line correctly in the space provided. If the line is correct, write a "C" in the space beneath the program line. (Finding errors in program lines is called "debugging" the program.) Each correct answer is worth three points.

1. 280 INPUT "ENTER 1,2,3, OR 4 :;NU

2. 2030 IF N>0 THEN HOME, PRINT "YOUR NUMBER IS ";N :GOTO 2010

3. 3020 INPUT "ENTER TWO NUMBERS : ";AB

4. 1020 INPUT CHR$(7)

5. 250 IF W$ = "STOP" THEN RETURN 9000

6. 10 MULTIPLE CHOICE EXAMPLE

7. 4030 PRINT WHICH CLASS IS YOUR FAVORITE?

8. 570 PRINT TAB 10 ;"NAME OF SCHOOL"

9. 1000 INPUT "ENTER A NUMBER BETWEEN 1 AND 4: ";N
   1010 IF N>1 OR N<4 THEN PRINT "WRONG NUMBER"
   1020 IF N>0 AND N<5 THEN PRINT "RIGHT NUMBER"

10. 1100 If N<>4 THEN 1060

Evaluation Test #4
Lessons Seven And Eight

DIRECTIONS — This test is divided into three parts. The directions for each part are explained separately. Total points: 100.

PART ONE DIRECTIONS — Choose the answer that BEST completes the statements and write the letter of your choice in the space provided. Each correct answer is worth 5 points.

1. ——— All of these statements correctly refer to an accumulator except

   A. The accumulator is a variable which counts the number of times a loop repeats itself.
   B. A name must be designated for the accumulator.
   C. The accumulator describes a special kind of variable.
   D. When placed in a loop, the value of the accumulator will change with each pass.

2. ——— Designing a program on paper is

   A. done in a precise and particular manner.
   B. something very few programmers do.
   C. done after the program is completed for documentation.
   D. an important habit to develop when programming.

3. ——— In this line:

   1000 ON A GOSUB 7000, 8000

   what is the computer being directed to do?

   A. If A, execute the subroutine at line 7000 and 8000.
   B. If A equals one, execute subroutine starting at line 7000. If A equals two, execute subroutine starting at line 8000.
   C. If A equals one, execute subroutine starting at line 7000 until line 8000 is reached.
   D. If A equals 7000, execute a subroutine. If A equals 8000, execute a subroutine.

4. ——— In the statement line:

   1000 FOR I =1 TO 10

   what is the "FOR" doing?

   A. Establishing the beginning and ending value of I.
   B. Establishing that I is equal to something between 1 and 10 depending on the INPUT from the operator.
   C. Establishing that I is equal to 11 because this computer is a base zero machine.
   D. Establishing a situation that will end a subroutine loop.

5. ——— The simplest means of establishing a loop is

   A. to document what you are going to do in the housekeeping section.
   B. to avoid nested loops whenever you can.
   C. to step by five instead of one because it executes more quickly.
   D. combining the functions of the accumulator and the IF—THEN command by using the FOR—NEXT paired command.

PART TWO DIRECTIONS — In this section you are asked to write, debug, and explain program lines. Each question is treated separately for point value.

1. Write the program lines that will write the message "PROGRAMMING" on the screen for 10 seconds. Place the message five lines down on the screen beginning in position twelve. This question is worth 18 points.

2. Explain what is happening in each of these programming lines. This question is worth 12 points.

```
1000 HOME
1010 FOR I =1 TO 21 STEP 2
1020 VTAB I
1030 PRINT "WHAT'S HAPPENING "
1040 NEXT I
1050 END
```

3. At the end of this program, what will the value of the variable C be? This question is worth 3 points.

```
1030 C =0
1040 FOR I = 1 TO 5
1050 FOR K = 1 TO 10
1060 C = C + 1
1070 NEXT K
1080 NEXT I
1090 PRINT "LOOP COMPLETED"
1100 END
```

4. Find the errors in these programming lines. This question is worth 8 points.

```
1000 FOR I = 10 TO 20 STEP 2
1010 HOME
1020 VTAB : HTAB
1030 PRINT I; "LOOPS ARE FUN ";CHR$(7)
1040 FOR J = 1 TO 5
1050 NEXT I
1060 NEXT J
1070 HOME
1080 END
```

5. Find the errors. This question is worth 4 points.

```
2000 GOSUB 6000
2010 HOME
2030 GOTO 9000
6000 PRINT "BEGINNING OF SUBROUTINE"
6010 GOTO 2010
9000 END
```

PART THREE DIRECTIONS — Programmers sometimes use the technique of flowcharting to design their programs on paper. Five of the symbols and what they represent were covered in this course. In the space provided, draw each of the five symbols and briefly explain their function. Each answer is worth 6 points — 3 points for the correctly drawn symbol, and 3 points for the correct explanation.

1.

2.

3.

4.

5.

Evaluation Test #5
Lessons Nine And Ten

DIRECTIONS — This test is divided into three parts. The directions for each part are explained separately. Total points: 100.

PART ONE DIRECTIONS — Write the correct answer in the space provided. Each correct answer is worth 3 points.

1. ———————— This statement lists the information which will be placed into variables by the READ command.
2. ———————— This could be used to access each element of a multi-dimensional array.
3. ———————— A string of characters within quotes.
4. ———————— This is a string with no value.
5. ———————— This command assigns information to variables from a DATA statement.
6. ———————— A single subscripted variable from an array is referred to as this.
7. ———————— Used to convert a character into a number.
8. ———————— Takes a number and converts it into a character string.
9. ———————— Numbering an element within an array is referred to as this.
10. ———————— This is the most frequently used of the string manipulation functions.
11. ———————— The READ statement for an array is usually handled with this.
12. ———————— Putting any two pieces of a string together using a plus sign is called this.

PART TWO DIRECTIONS — Choose the BEST answer for each question and put the letter of your choice in the space provided. Each correct answer is worth 5 points.

1. ——— When assigning months to variables, the variables MA$, MB$, MC$ were used for the 10th,11th, and 12th months instead of M10$, M11$, and M12$ because

   A. the computer would read M10$, M11$, M12$ as the same variable because it recognizes only the first two characters.
   B. the 10th, 11th, and 12th months have two digits and the variable names must be different for two digits.
   C. the variable name was already established as having three characters and it maintains the pattern.
   D. when assigning character names, you can use only one number.

2. ——— In this statement line:

DIM SE$(5,2)

the two numbers enclosed in the parentheses indicate that

A. two arrays are to be dimensioned named SE$; the first having 5 elements, the second having 2 elements.
B. there are five single dimension arrays in the multi-dimensional array. There are two elements in each array.
C. there is one multi-dimensional array named SE$ with the equivalent of six single dimensional arrays with three elements each.
D. the DIM numbers' only variable will be reassigned to another variable which is a character variable.

3. ——— The command

PRINT MID$ (B$,4,3)

instructs the computer to

A. print four characters after the third character.
B. print the character midway between B$ and three, which is four.
C. begin with the fourth character of the string B$ and print three characters.
D. print MID$, then B$, then four, then three.

4. ——— The DIM command is used to

A. instruct the computer to dim the letters being printed on the screen so they will not be so bright.
B. instruct the computer to cause the size of the subscript on the screen to get smaller and smaller.
C. instruct the computer what the smallest subscript will be and in what order the elements will be printed.
D. show the computer what the largest subscript will be within the array.

5. ——— Two different arrays can be dimensioned on the same line if you

A. explain in the housekeeping section that you will be putting two different arrays on the same line.
B. leave a space between the array names.
C. separate the array names with a comma.
D. put a semicolon before the first array name.

PART THREE DIRECTIONS — In this section you will be writing, debugging, and interpreting program lines. Each question is treated separately for point value.

1. Debug (find the errors in) these program statements. 8 points.

```
200 DATA RED, BLUE, GREEN, BROWN, COLORS
210 READ C1$, C2$, C3$, C4$ C5$, C6$
```

2. Assume that the date was assigned to a variable like this:

```
D$ = "03211963"
```

If you wish to extract the year 1963, you could use two string manipulation functions. Write the programming lines that would show both examples. 8 points.

3. Debug these programming lines. 15 points.

```
1000 HOME
1010 INPUT "ENTER A WORD (/ TO END) :;W$
1020 IF W$ = "/" THEN 1000
1030 VTAB
1040 L=LEN (W$)
1050 PRINT W$;" HAS ";L;" CHARACTERS"
2030 VTAB 10
2040 INPUT "PRESS RETURN FOR NEXT WORD ";
2050 GOTO 1010
9000 END
```

4. Write what the following program lines will print on the screen. 4 points.

```
1000 D$ ="09101940"
1010 C$ = LEFT$(D$,2) + "/" + MID$(D$,3,2) + "/" + RIGHT$(D$,4)
1020 PRINT C$
```

5. Using the LEN function, how many characters would this statement have? 4 points.

"WE CAN ALL BE PROGRAMMERS"

Evaluation Test #6
Lessons Eleven And Twelve

DIRECTIONS — This test is divided into two parts. Each part is explained separately. Total points: 100.

PART ONE DIRECTIONS — Write the correct answer in the space provided. Each correct answer is worth two points.

1. ———————————— Command that sets the color to be used in lo-resolution graphics mode.

2. ———————————— Command that places a numeric value at a specified address in the computer's memory.

3. ———————————— Command that instructs the computer to generate a random number between zero and one.

4. ———————————— Command that sets the screen to lo-resolution graphics mode.

5. ———————————— Command that removes the decimal portion of a number.

6. ———————————— Command that sets the screen back to text mode.

7. ———————————— Command that draws a vertical line in lo-resolution graphics mode from one specified point to another at a given column.

8. ———————————— The regularly displayed screen which is divided into 24 lines and 40 columns.

9. ———————————— Command that causes the computer to read the numeric value at a specified address in the computer's memory.

10. ———————————— Command that instructs the computer to place a block of color at a designated row and column in lo-resolution graphics mode.

11. ———————————— The number of colors allowed in lo-res graphics.

12. ———————————— What specified memory locations are called.

13. ———————————— The area in which text is displayed on the CRT while in lo-res graphics.

14. ———————————— Command that instructs the computer to find the color of a specified location on the lo-resolution screen.

15. ———————————— The number of plotting positions on the lo-resolution screen.

16. ———————————— The command that draws a horizontal line in lo-resolution graphics mode from one specified point to another at a given row.

PART TWO DIRECTIONS — You will be explaining and writing program lines for this section. Each question is treated separately for point value.

1. Explain the following lines. Each correct explanation is worth 2 points.

```
1100 FOR I = 1 TO 15
1110 PRINT INT (4 * RND (1) +1)
1120 NEXT I
```

2. Using the memory address (-16384), write a program that will PEEK at the keyboard address and store the contents in the variable AB. Then instruct the computer to print the value of the variable. Finally, instruct the computer to repeat the process by forming an eternal loop. 12 points.

3. Write the program lines to cover the following problem. Show remarks to explain the nature of the program and a save line for program "TEST". Clear the screen. Put the computer in lo-res graphics mode. Position the cursor on text line 22 at position one. Create a prompt for the column coordinate and store in variable "X". This is to be the horizontal coordinate in the plot command. Instruct the operator to enter a minus one to end the program. Then check to see if the operator has entered a minus one to end the program. If a minus one has been entered, then jump to line 9000 and end the program. Write a line to see if the allowable graphics range of zero to 39 has been entered. If not, then ask for another horizontal coordinate. End the program lines. This question is worth 50 points.

Evaluation Test #7

Lessons Thirteen And Fourteen

DIRECTIONS — This test is divided into two parts. Each part is explained separately. Total points: 100.

PART ONE DIRECTIONS — Write the correct answer in the space provided. Each correct answer is worth three points.

1. ———————— A data file used by the computer in which the information is stored in one long connected file.

2. ———————— Command that instructs the computer to place the value of a single character from the keyboard into a character variable.

3. ———————— Once the cursor is positioned, these two keys allow you to erase the rest of a line.

4. ———————— A DOS command which instructs the computer to store information in a data file.

5. ———————— Command that instructs the computer to return to the first data statement line within a program for use with the newest READ command.

6. ———————— A diskette file used for the permanent storage of information.

7. ———————— A space of defined length; used in random access files for holding related information.

8. ———————— If you hit this key once, you will be in edit mode.

9. ———————— Command that instructs the computer to stop displaying line numbers during execution.

10. ———————— A data file used by the computer to store information in the order specified by the program.

11. ———————— The escape key and this key will allow you to erase from the cursor to the end of the screen.

12. ———————— A DOS command which instructs the computer to ready a data file for information storage and retrieval.

13. ———————— A portion of the screen that the text lines or lines of information to the operator will move through, while the rest of the screen remains stationary.

14. ———————— The command that is used to set what is called an error trap. It is used to keep the program from halting when incorrect information is entered.

15. ———————— Command that executes a machine language subroutine from a BASIC program.

16. ———————— A DOS command which prevents data transfer and protects a previously opened data file.

17. ———————— A DOS command that instructs the computer to retrieve information from a data file.

18. ———————————— Cancels the line currently being entered or edited.

19. ———————————— Command that instructs the computer to display the program line number as it executes that line.

20. ———————————— Instructs the computer to add to a sequential file without writing over the information that already exists in the file.

PART TWO DIRECTIONS — In this section you will debug and write program lines.

1. Debug and rewrite the incorrect program lines below: 10 points.

```
 200 DIM N$(2),A(2)
1000 PRINT CHR$(4) ; OPEN AGES
1010 INPUT CHR$(4) ;"READ AGES"
1020 FOR I = 0 TO 2
1030 INPUT N$(1)
1040 INPUT A(2)
1050 NEXT I
1060 PRINT CHR$ ;"CLOSE AGES"
```

2. Write the appropriate programming lines for each of these statement: 30 points.

A. You wish to add to a sequential file named "NAMES".

B. You wish to close the file named "ADDRESSES".

C. Dimension two arrays; one for names, one for telephone numbers. Allow enough room for three passes of the computer.

D. You want to write to a random file called "NAMES".

E. You want to open a random file named "GRADES". Each record length is 10.

F. Write a line that asks the operator to enter the number of the record to be read. Tell the operator zero will end the program.

Answers For Evaluation Test #1

PART 1

1. UNLOCK
2. CURSOR
3. BASIC
4. INVERSE
5. NEW
6. CHARLES BABBAGE
7. CATALOG
8. BUG
9. REM
10. PROCESSOR
11. DISKETTE
12. DOS OR DISK OPERATING SYSTEM
13. SPEED = 255
14. PROGRAM
15. LOAD
16. CRT OR CATHODE RAY TUBE
17. LIST
18. INITIALIZATION
19. END
20. CTRL KEY
21. INIT
22. DELETE
23. SPEED
24. RUN
25. GREETING PROGRAM
26. HOME
27. FLASH
28. RENAME
29. PRINT
30. LOCK
31. STATEMENT LINES
32. DIRECT COMMAND
33. SAVE
34. ZERO (0)
35. CHARACTER

PART 2

1. SPEED = any number from 0 to 255

2-4. A. File name must begin with a letter.
B. Any typeable character, except a comma, may appear in the name.
C. File names must be from 1 to 30 characters in length.

5. RENAME current file name, new file name

6. An asterisk appears in front of locked files in a CATALOG listing.

7. LIST beginning line number, ending line number

8. PRINT "COMPUTER PROGRAMMING IS FUN"

9. REPT and "A"

10. A blank line will be printed.

Answers For Evaluation Test #2

PART 1

1. PROMPTED INPUT
2. TAB( )
3. ^
4. QUOTATION MARKS
5. LET
6. COMMA
7. $
8. +
9. VARIABLE
10. HTAB
11. /
12. CHARACTER STRING OR STRING LITERAL
13. INPUT
14. VARIABLE NAME
15. PROMPT
16. VTAB
17. *
18. SEMICOLON
19. -
20. VARIABLE NAME

PART 2

1. B
2. C
3. A
4. D
5. A
6. C
7. B
8. D

PART 3

LINE 500 — Set the PRINT speed to its slowest speed, zero.

LINE 1020 — The cursor will be moved to position 20.

LINE 1060 — The cursor is positioned to line 4.

LINE 1070 — The computer is instructed to begin at the current cursor position and print blanks up to the TAB position (position 5). The computer then prints "1ST TAB" on the same line because of the semicolon. Then the computer prints blanks from the current position up to the TAB position 25 and prints "2ND TAB".

LINE 1090 — The INVERSE command is cancelled and the computer returns to NORMAL print mode.

Answers For Evaluation Test #3

PART 1

1. Eternal Loop
2. PRINT CHR$(7)
3. <
4. AND
5. GOTO
6. Do an outline or logic paragraphs to plan your program
7. CTRL-S
8. IF-THEN
9. <>
10. Dummy INPUT
11. DEL
12. OR
13. >
14. ?
15. CTRL-C
16. Comma(s)
17. Multiple Input
18. CTRL-RESET
19. :
20. =

PART 2

1. If the value of the variable N is less than one or if the value of N is greater than 5, then print the statement "ENTER ANOTHER NUMBER" on the screen.

2. If the value of N is not equal to zero, jump to line 1000 and continue execution from there.

3. This is an INPUT statement asking for a "YES" or "NO" answer from the operator.

4. This will print on the screen the words "THE SCORE YOU ENTERED WAS " and whatever score was assigned to the variable S.

5. This is an example of a dummy input or stops the program flow until the operator presses RETURN.

6. This statement will produce the "bell" sound.

7. This assigns the "bell" sound to a variable and then produces the "bell" sound.

8. If the value of the variable G is equal to zero, then jump to line 4000 and continue execution from there.

9. This is a multiple statement line. First the screen will be cleared. Then the value of the variable BL$ will be printed. The cursor will then move to line 8 and place the CRT in the flashing mode.

10. This is a remark line and is ignored by the computer. It is for the programmer's benefit only.

PART 3

1. 280 INPUT "ENTER 1,2,3, OR 4 :";NU

   There was a missing quotation mark after the colon and before the semicolon.

2. 2030 IF N>0 THEN HOME : PRINT "YOUR NUMBER IS ";N : GOTO 2010.

   After the word HOME, there should have been a colon.

3. 3020 INPUT "ENTER TWO NUMBERS : ";A,B

   There should have been a comma between the A and the B.

4. 1020 PRINT CHR$(7)

   The word INPUT should have been PRINT.

5. 250 IF W$ = "STOP" THEN GOTO 9000

   OR

   250 IF W$ = "STOP" THEN 9000

   It should read GOTO after the word THEN rather than RETURN.

6. 10 REM MULTIPLE CHOICE EXAMPLE

   This line needed a REM statement at the beginning.

7. 4030 PRINT "WHICH CLASS IS YOUR FAVORITE ?"

   WHICH CLASS IS YOUR FAVORITE ? should have been enclosed in quotation marks.

8. 570 PRINT TAB(10); "NAME OF SCHOOL"

   The numeral 10 should have been enclosed in parentheses.

9. 1000 INPUT "ENTER A NUMBER BETWEEN 1 AND 4 : ";N
   1010 IF N<1 OR N>4 THEN PRINT "WRONG NUMBER"
   1020 IF N>0 OR N<5 THEN PRINT "RIGHT NUMBER"

   The "greater than" and "less than" symbols in line 1010 are reversed.

10. C. This line is correct.

Answers For Evaluation Test #4

PART 1

1. C
2. A or D
3. B
4. A
5. D

PART 2

1. 1000 HOME
   1010 VTAB 5
   1020 HTAB 12
   1030 PRINT "PROGRAMMING "
   1040 FOR I = 1 TO 10 * 740
   1050 NEXT I
   1060 HOME

2. Line 1000 clears the screen.

   Line 1010 — the FOR command establishes conditions for the loop. On the first pass of the loop the variable I will be equal to one. STEP 2 tells the computer to add 2 to the value of I each time the loop repeats. The ending of the loop will occur when I is equal to 21.

   Line 1020 positions the cursor on the screen line equal to the value of I. The first time it will be one, the second time three, and so on.

   Line 1030 "WHAT'S HAPPENING " will be printed at the line assigned by the VTAB.

   Line 1040 executes the next pass of the loop.

   Line 1050 ends the program.

3. 50

4. There were two errors.

   A. Line 1020 VTAB : HTAB
      Neither command specified a value.

   B. Line 1050 and line 1060 should have been reversed. (Always finish the inside loop first.)

5. The error is in line 6010. The line should read: 6010 RETURN

PART 3

1. Terminal (oval shape) — indicates beginning and ending of a program. Can also indicate beginning and ending of a subroutine.
2. Input/Output or I/O (parallelogram shape) — shows what will be entered (input) or what printed (output). will be printed (output).
3. Process Block (rectangle shape) — the most general of the symbols. Used to represent math operations, direct assignment of variables, or any other operation which does not correspond with any of the other four symbols.
4. Decision (diamond shape) — used with commands IF-THEN, FOR-NEXT, ON-GOSUB, and ON-GOTO.
5. Predefined Process (rectangle with lines drawn down either side) — represents a predefined process. Used in place of a series of programming lines which perform a single task (usually a subroutine).

Answers For Evaluation Test #5

PART 1

1. DATA STATEMENT
2. NESTED LOOP
3. STRING LITERAL, or CHARACTER STRING
4. NULL STRING
5. READ
6. ELEMENT
7. VAL
8. STR$
9. SUBSCRIPTING
10. MID$
11. FOR-NEXT LOOP
12. CONCATENATION

PART 2

1. A
2. C
3. C
4. D
5. C

PART 3

1. A. There is a comma missing between variables C4$ and C5$.
   B. There are more variables than data.

2. 500 PRINT MID$ (D$,5,4) or 500 PRINT MID$ (D$,5)
   510 PRINT RIGHT$ (D$,4)

3. LINE 1010 — Quotes missing after message statement.

   LINE 1020 — Should be: IF W$ = "/" THEN 9000

   LINE 1030 — No assigned number to VTAB

   LINE 2040 — Variable for dummy input is missing after semicolon.

4. 09/10/1940

5. 25 CHARACTERS. (Don't forget the last space.)

Answers For Evaluation Test #6

PART 1

1. COLOR =
2. POKE
3. RND( )
4. GR
5. INT( )
6. TEXT
7. VLIN
8. TEXT MODE OR TEXT SCREEN
9. PEEK( )
10. PLOT
11. 16
12. ADDRESSES
13. TEXT WINDOW
14. SCRN( )
15. 1600
16. HLIN

PART 2

1. LINE 1100 of the FOR-NEXT loop instructs the computer to assign the value of one to the variable I and increase it by one with each pass of the loop until it equals fifteen.

   LINE 1110 instructs the computer to print the integer portion of the results of the calculation. The computer will return only whole numbers between 1 and 4.

   LINE 1120 instructs the computer to execute the next pass of the loop.

2. 1000 AB = PEEK(-16384)
   1010 PRINT AB
   1020 GOTO 1000

3. 10 REM :: PGM FOR TEST
   20 REM SAVE TEST
   500 HOME
   510 GR
   1000 VTAB 22
   1010 INPUT "ENTER COLUMN COORDINATE (-1 TO END) : ";X
   1020 IF X = -1 THEN 9000
   1030 IF X<O OR X>39 THEN 1000
   9000 END

Answers For Evaluation Test #7

PART 1

1. SEQUENTIAL FILE
2. GET
3. ESC E
4. WRITE
5. RESTORE
6. DATA FILE OR TEXT FILE
7. RECORD
8. ESC KEY
9. NOTRACE
10. RANDOM ACCESS FILE
11. ESC F
12. OPEN
13. SCROLLING WINDOW
14. ONERR GOTO
15. CALL
16. CLOSE
17. READ
18. CTRL X
19. TRACE
20. APPEND

PART 2

1. LINE 1000 — OPEN AGES needs to be enclosed in quotes
   LINE 1010 — INPUT is incorrect. Should be PRINT
   LINE 1030 — I should be enclosed in the parentheses following the variable N$
   LINE 1040 — I should be enclosed in the parentheses following the variable A
   LINE 1060 — CHR$ should be followed by (4)

   The corrected program should look like this:

```
 200 DIM N$(2), A(2)
1000 PRINT CHR$(4) ;"OPEN AGES"
1010 PRINT CHR$(4) ;"READ AGES"
1020 FOR I = 0 TO 2
1030 INPUT N$(I)
1040 INPUT A(I)
1050 NEXT I
1060 PRINT CHR$(4) ;"CLOSE AGES"
```

2. A. 2000 PRINT CHR$(4) ;"APPEND NAMES"

   B. 2000 PRINT CHR$(4) ;"CLOSE ADDRESSES"

   C. 100 DIM N$(2), P(2)

   D. 1000 PRINT CHR$(4) ;"WRITE NAMES, R";N

   E. 2000 PRINT CHR$(4) ;"OPEN GRADES,L10"

   F. 2000 INPUT "ENTER RECORD NUMBER TO READ (0 TO END) ";N

VideoSoft™ 2101 South Broadway, Little Rock, Arkansas 72206